Design and Optimization of a Traffic Light Controller FSM

Ziya Kadir TOKLUOGLU
CSE 436/536: Term Project

January 14, 2025

Contents

1 Introduction

Finite State Machines (FSMs) are fundamental in digital system design, providing a structured approach to sequential logic implementation. This report details the design, simulation, and optimization of a traffic light controller FSM for a pedestrian crossing, implemented using Magic, SPICE, and the Logical Effort methodology.

2 FSM Design

The traffic light controller FSM consists of three primary states: Green, Yellow, and Red. The FSM operates based on input signals such as a pedestrian request (REQ), clock (CLK), and reset (RST). Outputs control the traffic light signals and pedestrian walk/stop indications.

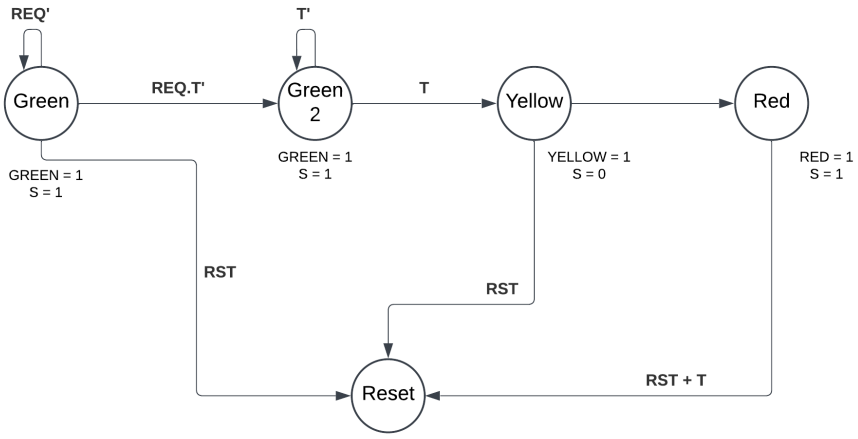

Figure 1: Finite State Machine for Traffic Light Controller

2.1 State Formulas

The following formulas represent the logic expressions for the state bits $N2$, $N1$, and $N0$ of the FSM:

$$N2 = \text{RST}$$

$$N1 = \Big[(\text{REQ} + S_0)' + \big((S_1' + T) + (S_0 + \text{REQ})'\big)'' + \big(S_1' + (T \cdot S_0)''\big)'\Big]'$$

$$N0 = \Big[(\text{REQ} + S_0)' + \big((S_1' + T') + (S_0 + \text{REQ})'\big)'' + \big(S_1' + (T \cdot S_0)''\big)'\Big]'$$

Additionally, the output logic expressions for the traffic light and pedestrian signal are defined as follows:

$$R = S_1 \cdot S_0$$

$$G = S_0'$$

$$Y = S_1 \cdot S_0'$$

$$S = (S_1 \cdot S_0' + S_2)'$$

2.1.1 Explanation

- $N2$**:** The RST signal solely determines the $N2$ state bit, ensuring all states transition to Reset when RST is asserted. This formula requires no additional gates as $N2 =$ RST.
- $N1$ **and** $N0$**:** These state bits depend on more complex conditions involving REQ, S_0, S_1, T, and their combinations. While the formulas for $N1$ and $N0$ are similar, slight differences necessitate individual logic representations. Shared logic is reused to minimize gate usage and improve efficiency.
  - $N1$ requires 11 gates.
  - $N0$ requires 12 gates.
  - $GREEN$) requires 1 gates.
  - RED) requires 2 gates.
  - $YELLOW$) requires 2 gates.
  - When the logic for $N1$ and $N0$ is combined, the total gate count is reduced to 20 gates due to the reuse of 3 common gates.
- **Output Signals:** The output signals R, G, Y, and S are calculated based on the state bits S_1, S_0, and S_2. Each signal corresponds to the respective light or pedestrian signal:
  - R: Red light, active when S_1 and S_0 are both high.

- G: Green light, active when S_0 is low.
- Y: Yellow light, active when S_1 is high and S_0 is low.
- S: Pedestrian signal, active based on a combination of S_1, S_0, and S_2.

2.1.2 Total Gate Count

The FSM representation requires a total of 25 gates without DFF(D-flipflop), with no additional gates needed for $N2$, as it is directly driven by the RST signal.

2.2 Logic Implementation

The FSM was implemented using D flip-flops for state memory and combinational logic gates for next-state and output logic. The circuit design follows a synchronous clocked approach to ensure reliable operation.

3 Simulation and Results

Functional verification of the FSM was performed using Ngspice. Waveforms validating the state transitions and output behavior under various input conditions are presented below.

3.1 N1 SPICE Simulation

The SPICE simulation for the $N1$ state bit was conducted to verify its behavior under all relevant input conditions. The simulation ensures correct logic implementation and gate functionality. The waveform below shows the output transitions of $N1$ based on the provided input signals.

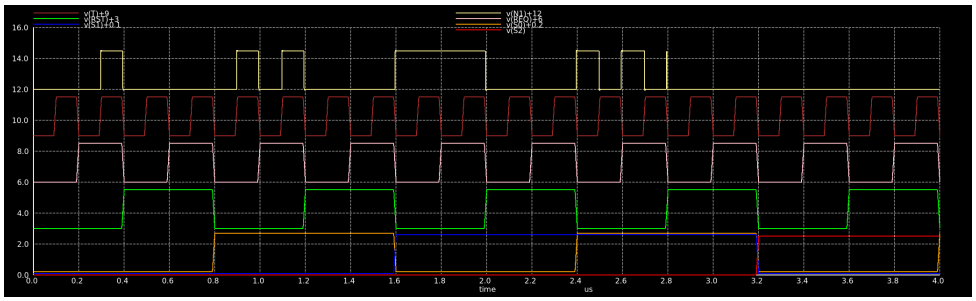

Figure 2: SPICE Simulation Waveform for $N1$

3.2 N1 and N2 SPICE Simulation

The combined simulation of $N1$ and $N2$ state bits was performed to validate their interaction and ensure accurate transitions. The shared logic between $N1$ and $N2$ was tested for efficiency and correctness. The waveform below demonstrates the combined output of $N1$ and $N2$ under various input conditions.

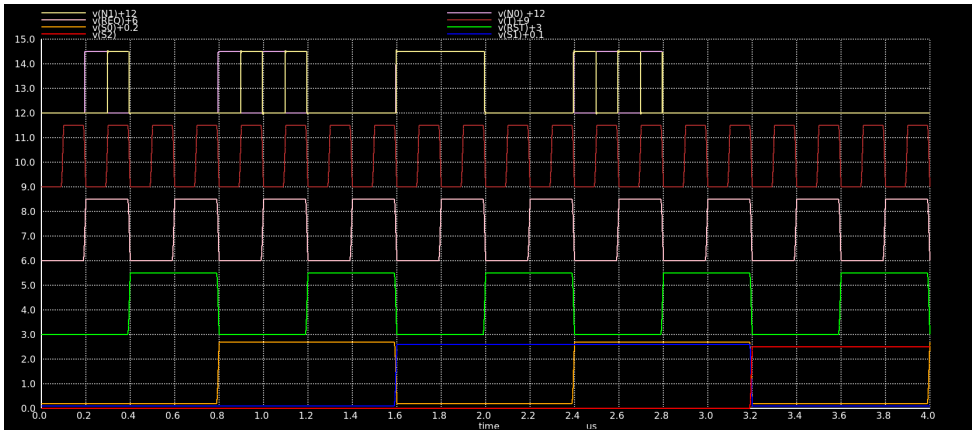

Figure 3: SPICE Simulation Waveform for $N1$ and $N2$

4 Logisim Design

The FSM design was implemented in Logisim to verify its logical correctness and simulate its behavior before proceeding to physical layout and SPICE simulations. The circuit includes state memory, implemented using D flip-flops, and combinational logic for next-state and output logic. The figure below illustrates the complete FSM design as created in Logisim, highlighting the modular approach and logical connections used in the implementation.

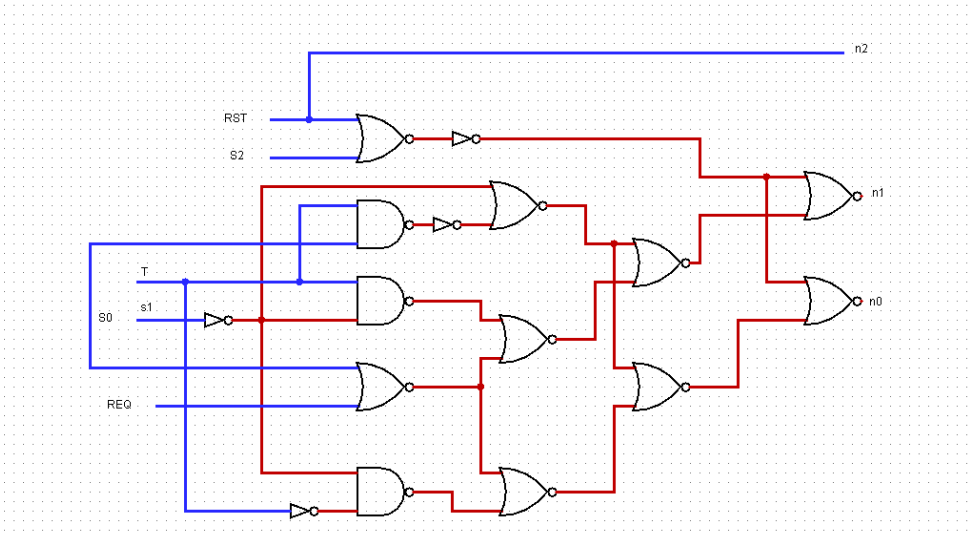

Figure 4: SPICE Simulation Waveform for $N1$ and $N2$

5 Conclusion

After completing all the FSM design and associated operations, the individual components were integrated into the full system. However, during the initial integration phase, the output was not as expected. Upon further refinement and adjustments, the final output, as shown below, was successfully achieved, demonstrating the FSM's intended functionality.

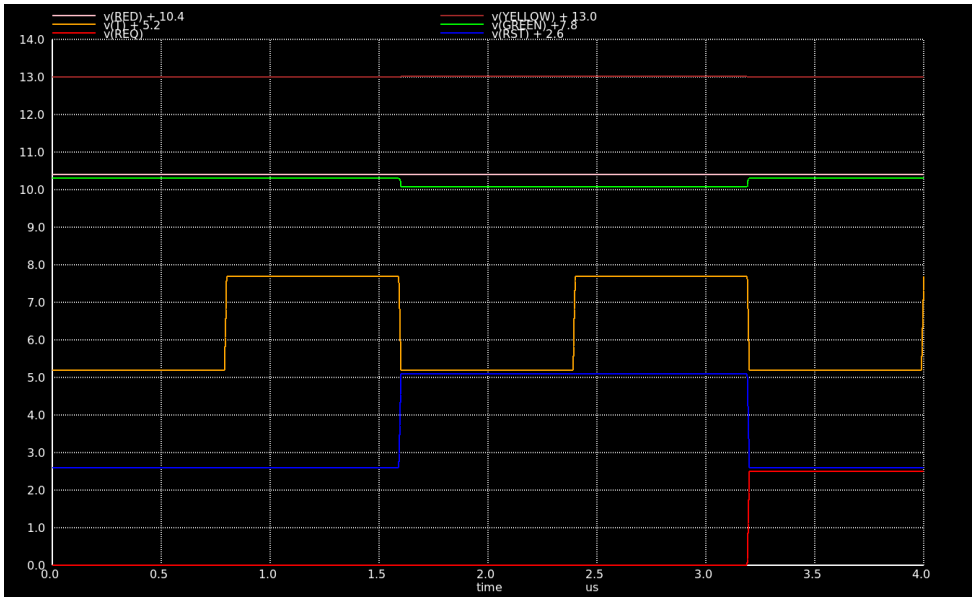

Figure 5: SPICE Simulation Waveform for $N1$ and $N2$