Princess Sumaya University for Technology

King Abdullah II Faculty of Engineering

Electrical Engineering Department

Princess Sumaya University for Technology جامعة الأميرة سميّة للتكنولوجيا

Digital Logic Lab
Final Project
A Traffic lights controller for a T-intersection.

Authors:			*Supervisor:*
Albert Saman	20220328	Electronics Eng.	Eng. Raghad
Hussam Dawood	20220168	Electrical power Eng.	

Saturday, May 25, 2024

Abstract

This report presents the design and simulation of a traffic light controller for a T-intersection using Verilog. The model aims to optimize traffic flow and enhance pedestrian safety through efficient signal management by coordinating vehicular and pedestrian traffic at the intersection.

TABLE OF CONTENTS

1 Introduction

In this report, we will define the designated T-intersection and a model solution for a traffic controller to control the motion of vehicles and pedestrians.

1. Objectives

The primary objective of this report is to design and simulate a traffic controller for a T-intersection that has four distinct modes, each with a different time interval. The intersection joins three roads (A, B, C) as well as a pedestrian area.

2. Theory

The T-intersection is given as shown in Figure 1.1.

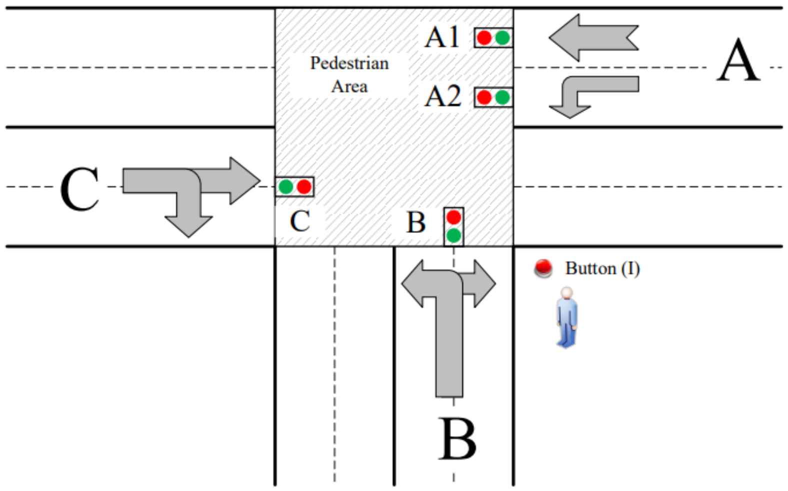


Figure 1.1: T-intersection Diagram

We notice the three-road connection and the pedestrian area. Road A has two lanes: A1 and A2 each with a different traffic light. Traffic of Lane A1 moves straight ahead while traffic of lane A2 moves to the left. Road B has two lanes with one traffic light. Traffic in road B can turn left or right only. Road C also has two lanes with one traffic light. Traffic in road C can move straight ahead or to the right.

There are four modes for this T-intersection with varying durations which will be discussed in more detail later. For now, we have to know that these modes will take different amounts of time.

We also see that there is a button (I) that can be pressed by pedestrians to allow them to enter the pedestrian area and cross the road.

The conditions and requirements of the design solution are described as follows:

- The traffic lights at this intersection operate with **only two signals**: 1 for Green and 0 for Red. (i.e., no Yellow).
- The sequence of traffic lights should follow (mode 1, mode 2, mode 3, mode 1, mode 2, etc…)
- When the pedestrian button is pressed, the controller should transition to **mode 0 after completing the current mode time.**
- Once the time for mode 0 has elapsed, the controller **will resume with mode 1**, mode 2, mode 3, etc...
- Assume that the system runs at a 1 Hz clock.
- If you need to create a counter, then it should be designed **structurally**.

--Solution--

After brainstorming, we settled on the idea of a counter that increments by one every positive edge. The value of the counter will be compared with the required period for each state and if they match, the mode changes to the succeeding state according to the input (I). The state diagram of the modes that we created is shown in Figure 1.2.

The top module was mainly done behaviorally and we instantiated the counter before the always loop. We made sure that the counter starts initially with a value of zero by initializing the JK flip-flops to zero in their module. Thus, when the flip-flops are instantiated in the counter module; they will begin with a value of zero.

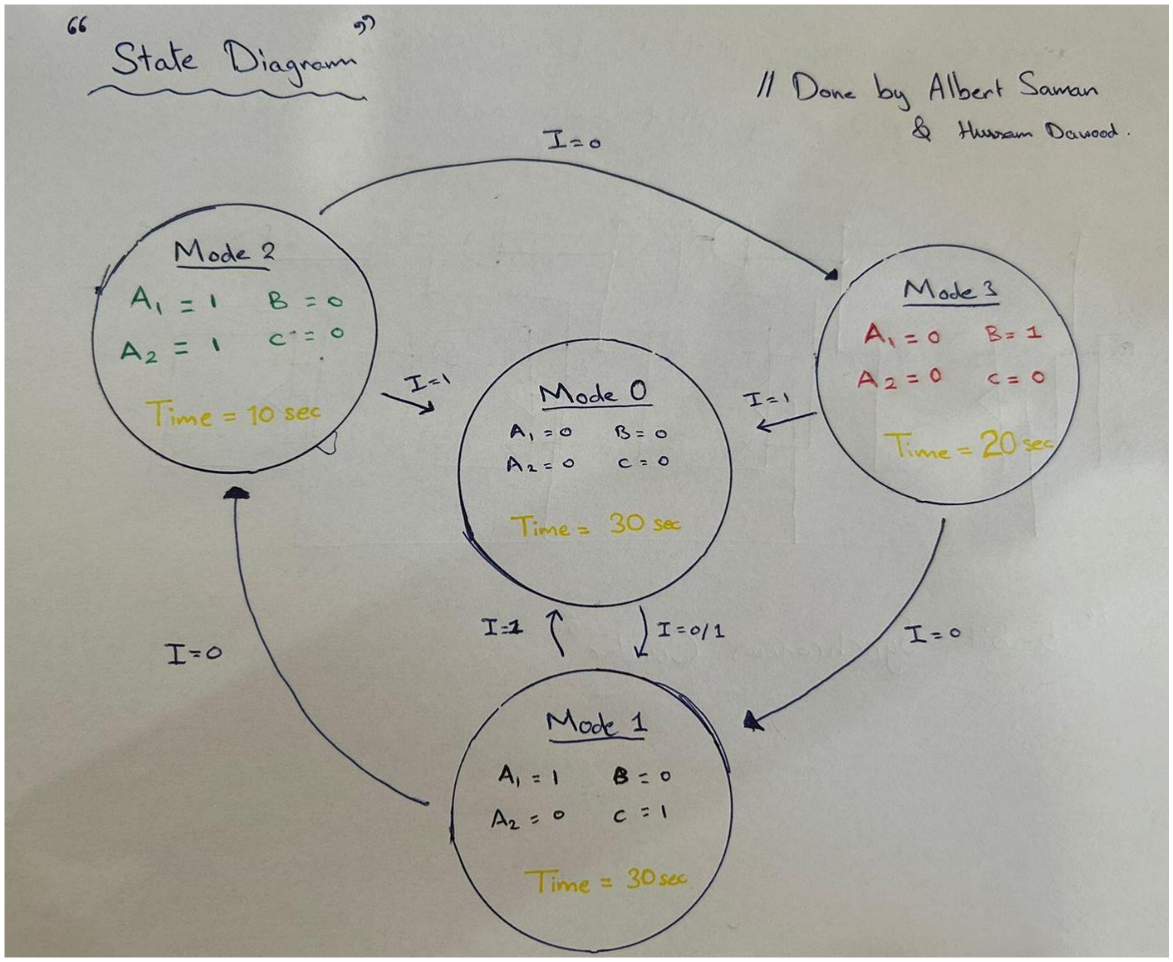


Figure 1.2: State Diagram

The body of the top module consists of a case statement for the mode. At every positive edge, the counter is checked in the mode we are currently in to see if it has terminated. If it did end, the counter clears (returns to zero), the states of the next mode are loaded in the outputs, and the mode is changed to the following mode. The next mode is determined from the value of the input I. If I = 0 (button was not clicked) then the next mode will be 1, 2, or 3 depending on the current mode. However, if I = 1 (button was clicked) then the next mode has to be 0.

One major point that needs attention is to check if clear is on at the beginning of every positive edge. If it is on, this means that we have just come from a mode change and we now need to start counting again. Therefore, clear is given a value of 0 again to stop it from clearing the counter. This idea is demonstrated in the diagram of Figure 1.3.

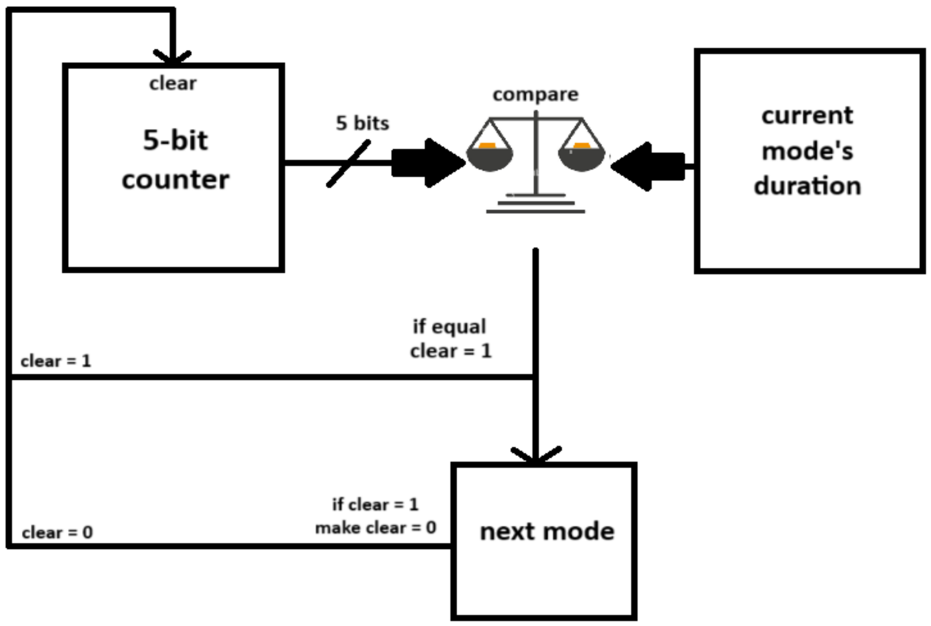


Figure 1.3: Idea Behind the Clear Register

Finally, in the test bench, we have to make the delay double the value of the time. For example, if the time is 10 seconds, then the delay must be 20 time units (#20). This is because at 1 Hz frequency, the period is 1 second and we know that 1 period has 2 time units. So, we would need twice as many time units as the time period for every mode. Figure 1.4 demonstrates the idea.

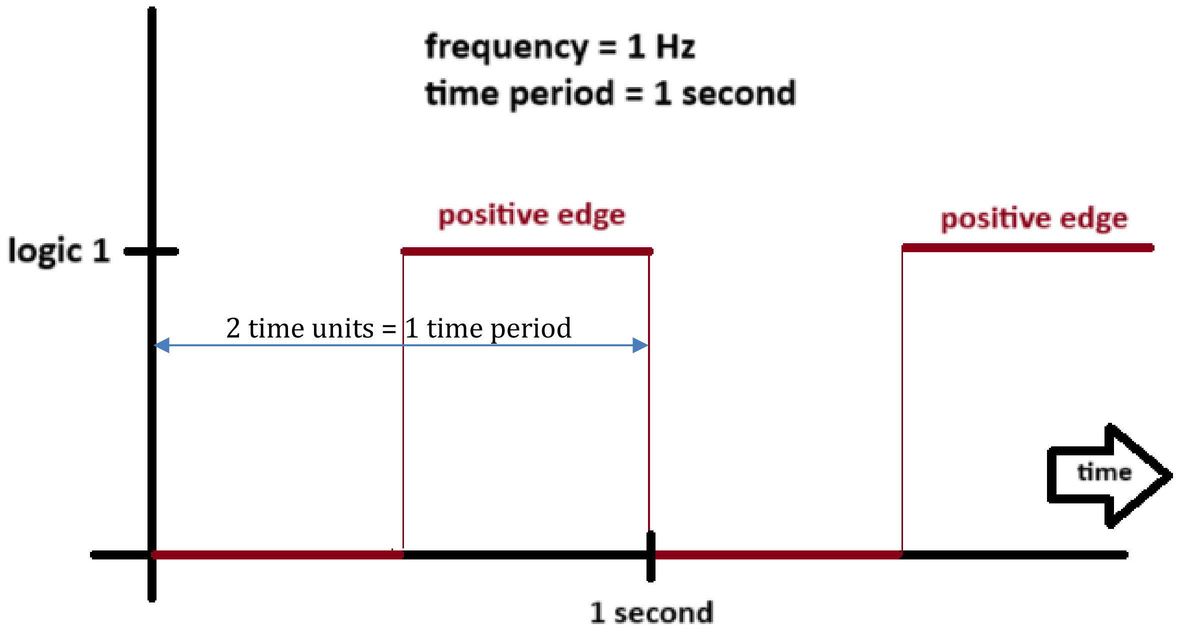


Figure 1.4: Clock Mechanism

2 PROCEDURE AND METHODS

As was said before, there are four different modes of traffic in the intersection with distinct time intervals as can be shown in Table 1 below where 0 means off and 1 means on.

Table 1: The modes of the intersection

Mode	A1	A2	B	C	I pressed	Period (seconds)
0	0	0	0	0	1	30
1	1	0	0	1	0	30
2	1	1	0	0	0	10
3	0	0	1	0	0	20

To implement the traffic light controller, a synchronous 5-bit counter was employed to manage the time intervals. We built the counter structurally using the Verilog. The output of the counter is 5 bits, allowing it to count from 0 to a maximum of 31 (11111 in binary) which is in the sufficient counting interval. This is because the highest count required by the designed circuit is 29 (11101 in binary). To achieve this, we used the following components to build the counter:

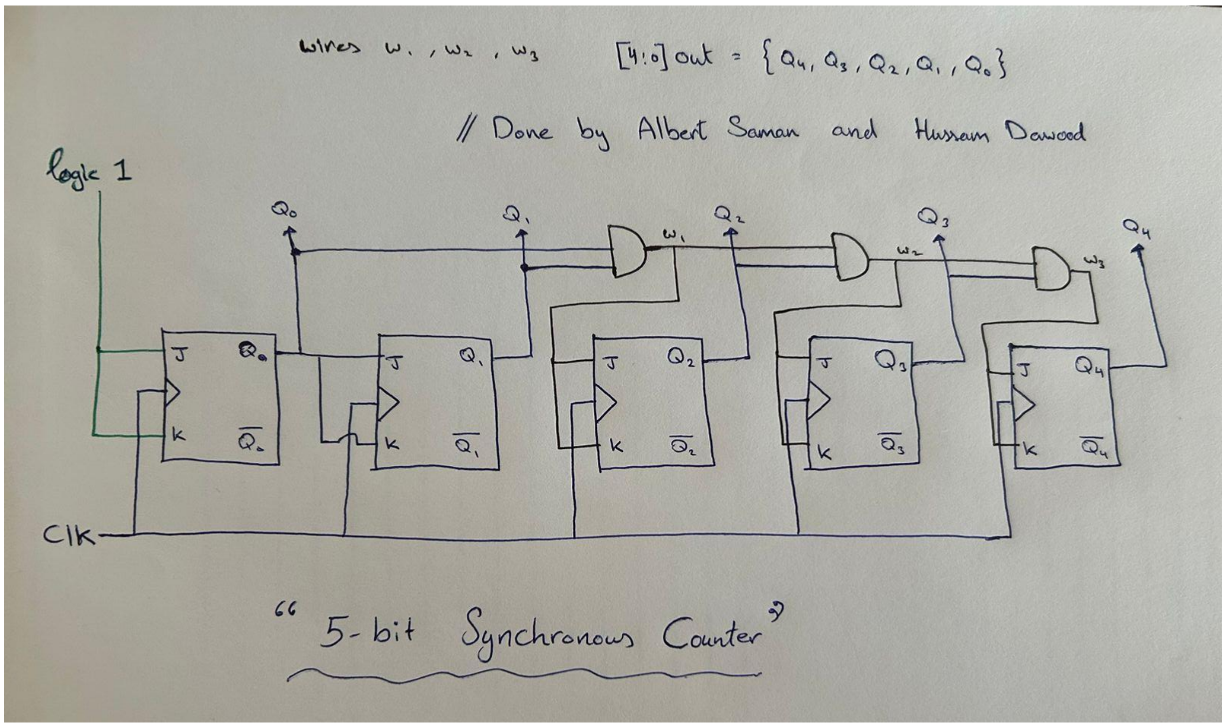

wires w₁ , w₂ , w₃
[4:0] out = {Q₄, Q₃, Q₂, Q₁, Q₀}
// Done by Albert Saman and Hussam Dawood
logic 1
Q₀
Q₁
Q₂
Q₃
Q₄
w₁
w₂
w₃
J
K
Clk
"5-bit Synchronous Counter"

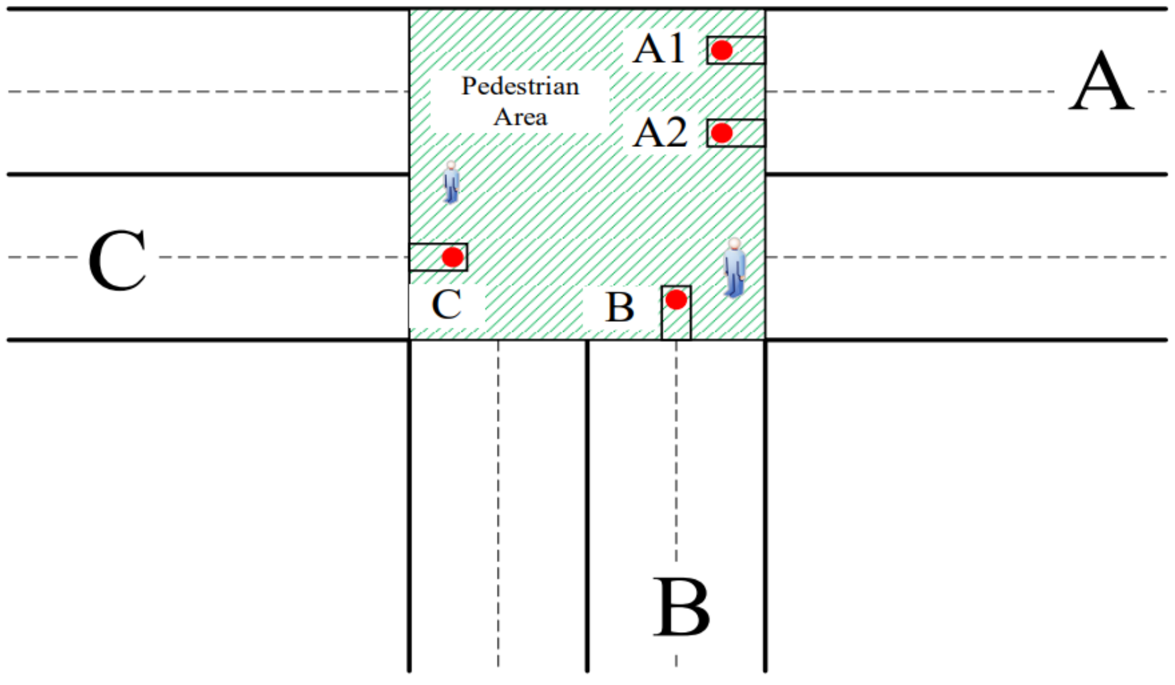
Figure 2.1: 5-bit Synchronous Counter Schematic

1. Five JK flip-flops to represent the 5 bits.
2. Three AND gates to ensure the design of the counter.
3. Logic one input
4. A clock

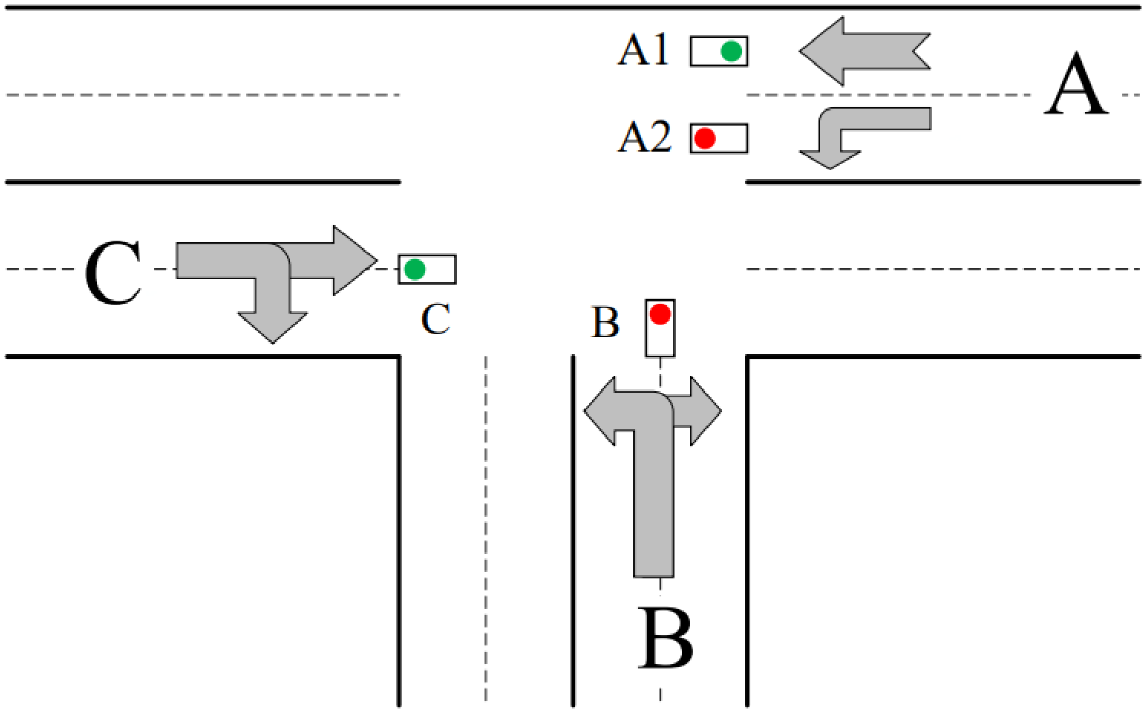
This setup enables precise control over the time intervals for each mode. As can be seen in Figure 2.1.

When implementing the counter in the main module, we took into account that the counter starts counting from zero. As a result, we would compare the value of the counter to the value of the mode duration minus one. For example, if we are in mode 1, the specified duration is 30 seconds. Therefore, we should compare the value of 29 to the counter output because the counter has already counted zero.

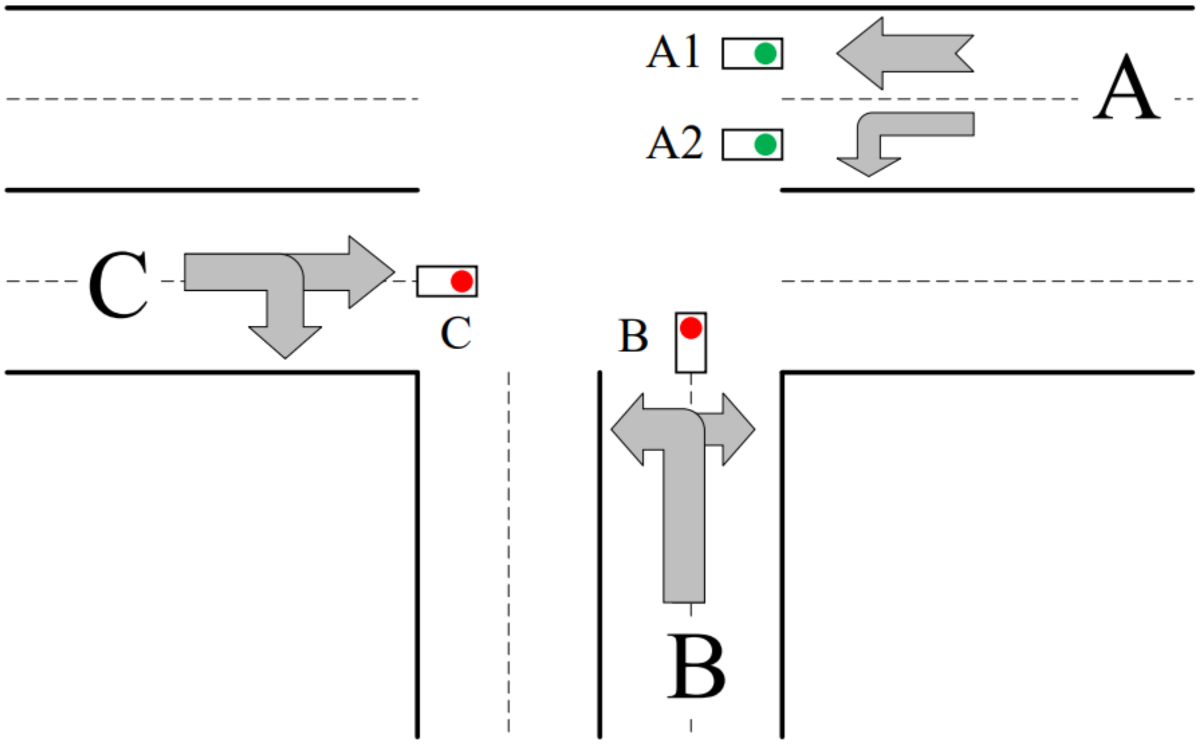
The four states required are shown in Figure 2.2.



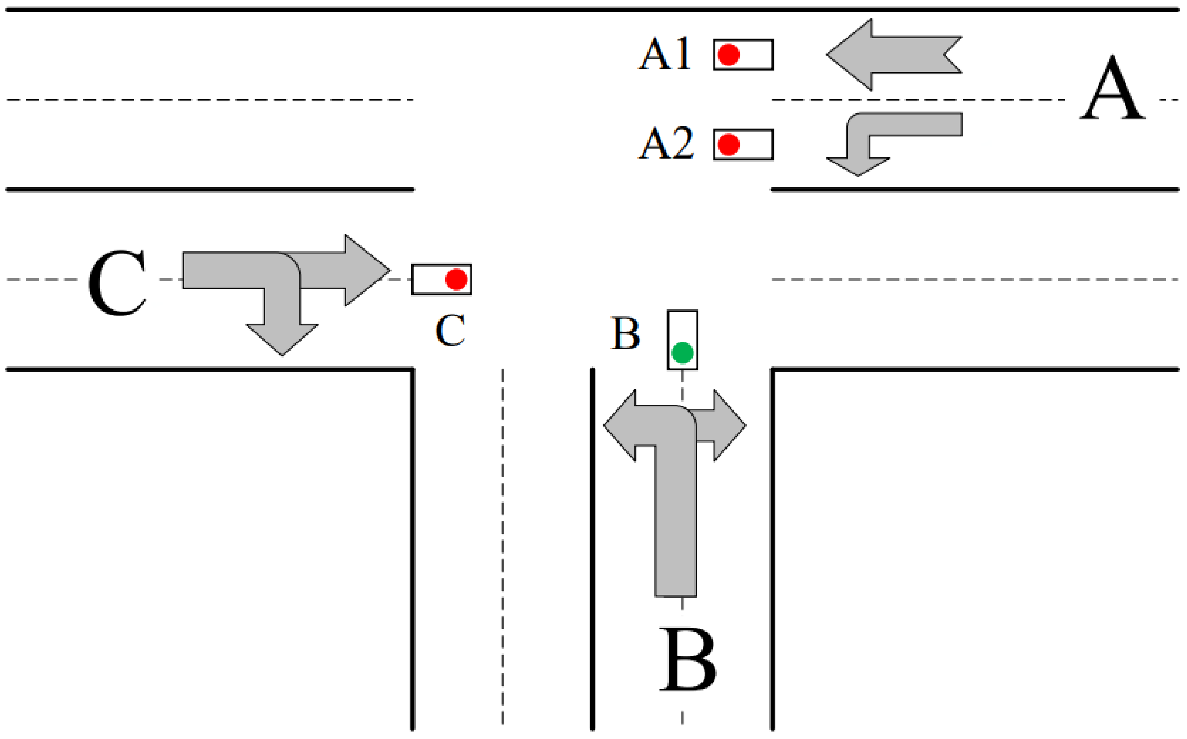
Mode 0



Mode 1



Mode 2



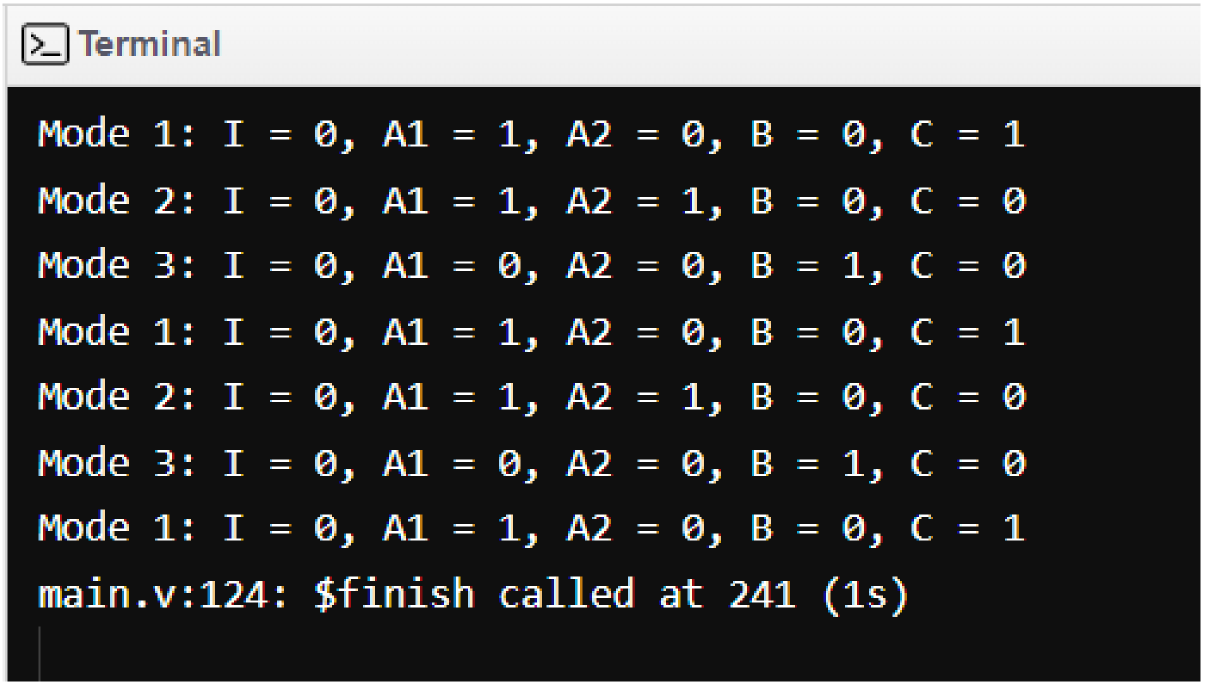
Mode 3

Figure 2.2: Traffic Modes

3 Results and Discussions

To discuss the report's results in detail, two test benches were made to understand the system's behavior with and without the pedestrian button activated.

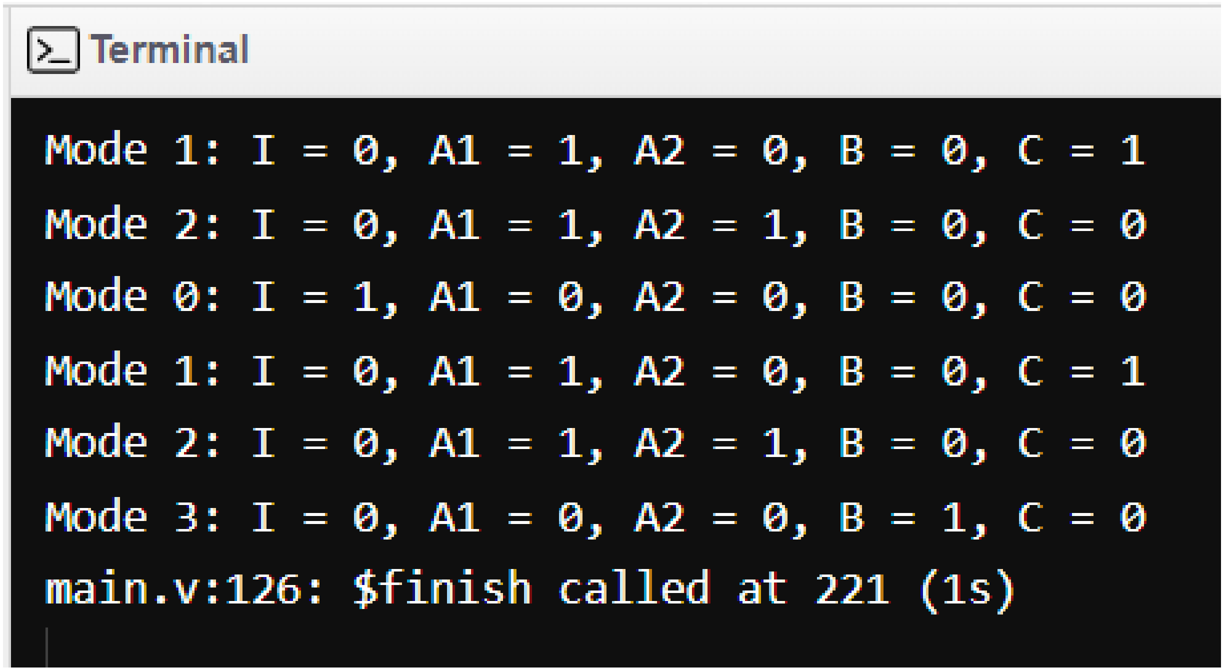
In the first test bench (Figure 3.1), the system cycled through modes 1 to 3, repeating these modes with their respective time intervals as long as the pedestrian button remained untouched (I=0).


Terminal

```
Mode 1: I = 0, A1 = 1, A2 = 0, B = 0, C = 1
Mode 2: I = 0, A1 = 1, A2 = 1, B = 0, C = 0
Mode 3: I = 0, A1 = 0, A2 = 0, B = 1, C = 0
Mode 1: I = 0, A1 = 1, A2 = 0, B = 0, C = 1
Mode 2: I = 0, A1 = 1, A2 = 1, B = 0, C = 0
Mode 3: I = 0, A1 = 0, A2 = 0, B = 1, C = 0
Mode 1: I = 0, A1 = 1, A2 = 0, B = 0, C = 1
main.v:124: $finish called at 241 (1s)
```

Figure 3.1: Operation of the model when the Button is not pressed (I = 0)

In the second test bench (Figure 3.2), the system began in mode 1 and transitioned to mode 2. However, after pressing the pedestrian button (I=1) after mode 2 had begun, the system reverted to mode 0 (after completion of mode 2, of course) to accommodate pedestrian crossing. After the allocated time for pedestrians to cross had finished, the system went back to cycle from mode 1 to mode 3 as long as the pedestrian button remained untouched (I=0). This behavior confirmed the controller's capability to prioritize pedestrian safety while managing the regular traffic flow efficiently.



```
Mode 1: I = 0, A1 = 1, A2 = 0, B = 0, C = 1
Mode 2: I = 0, A1 = 1, A2 = 1, B = 0, C = 0
Mode 0: I = 1, A1 = 0, A2 = 0, B = 0, C = 0
Mode 1: I = 0, A1 = 1, A2 = 0, B = 0, C = 1
Mode 2: I = 0, A1 = 1, A2 = 1, B = 0, C = 0
Mode 3: I = 0, A1 = 0, A2 = 0, B = 1, C = 0
main.v:126: $finish called at 221 (1s)
```

Figure 3.2: Operation of the Model when the Button is Pressed (I = 1)

4 CONCLUSIONS

To sum up, this report has detailed the design, implementation, and testing of a traffic light controller for a T-intersection using Verilog. The primary objective was to control traffic flow and enhance pedestrian safety. The controller was designed with four distinct modes with specific time intervals and included a mechanism to accommodate pedestrian crossings.

The implementation needed a synchronous 5-bit counter built structurally using JK flip-flops and AND gates to control time intervals. Through simulation and testing, the behavior was smoothly cycling through the modes and correctly transitioning for pedestrian crossings when needed.

To sum up, the traffic light controller effectively manages traffic and improves safety at a T-intersection for vehicles and pedestrians.

5 REFERENCES

[1]	PowerPoint presentation slides of Week 8 in digital logic lab