Princess Sumaya University for Technology

King Abdullah II Faculty of Engineering

Electrical Engineering Department

Digital Logic Lab 22348

Final Project - Spring 2024

A Traffic Lights Controller for a T-Intersection

Authors:			*Supervisor:*
Leen Amr	20210258	NIS Engineering	Dr. Amjad Al-Mousa
Mohammad Abdalaziz	20200168	Computer Engineering	

May 25, 2024

Abstract

A design and simulation of a traffic controller for a T-intersection aimed to optimize traffic flow and guarantee pedestrian safety using Verilog is presented in this project. The controller, with one input and four outputs, directs the movement of vehicles from three sides by having a two layered control system in one direction, even if it has a single input. There is also an additional feature which interrupts vehicular flow using pedestrian push button to facilitate crossings by foot. By simulating its operation, efficiency in handling vehicular movements and enhancing safety of pedestrians is confirmed leading to better intersection management systems.

Table of Contents

List of Figures

List of Tables

1-Introduction

The aim of this project is to simulate a traffic controller for a T-intersection using Verilog. Verilog is a hardware description language that is commonly used to design digital circuits. The designed controller operates by taking in information about the traffic and use it to decide when to turn the traffic lights on and off. The intersection has three roads coming in from different directions: A, B, and C. The traffic lights for roads B and C are controlled separately, while the traffic lights for road A are controlled by two different lights: A1, and A2. The intersection integrates a pedestrian push button (I) designed to stop all cars from passing enabling pedestrians to traverse the intersection safely. The code will analyze the traffic data and make decisions about when to turn the lights on and off to keep traffic moving smoothly.

1.1 Objectives:

- Using Verilog to implemet and design a traffic controller to manage flow at a T-intersection.
- Simulate the controller under normal and pedestrian crossing conditions to ensure safety and accessibility.
- Test the controller's performance to verify its response to changing traffic and pedestrian safety needs.

1.2 Components:

- A Verilog program: This is the code that implements the traffic signal controller.
- A sequence of traffic modes: This is the order in which the traffic signals in the Verilog program are activated to control the movement of vehicles and pedestrians.
- A mechanism for handling pedestrian movements: This is likely a separate set of controls in the Verilog program that allow pedestrians to safely cross the intersection by controlling the traffic light.

2- Design and Functionality of the Traffic Controller

This traffic light controller has 4 modes as the following:

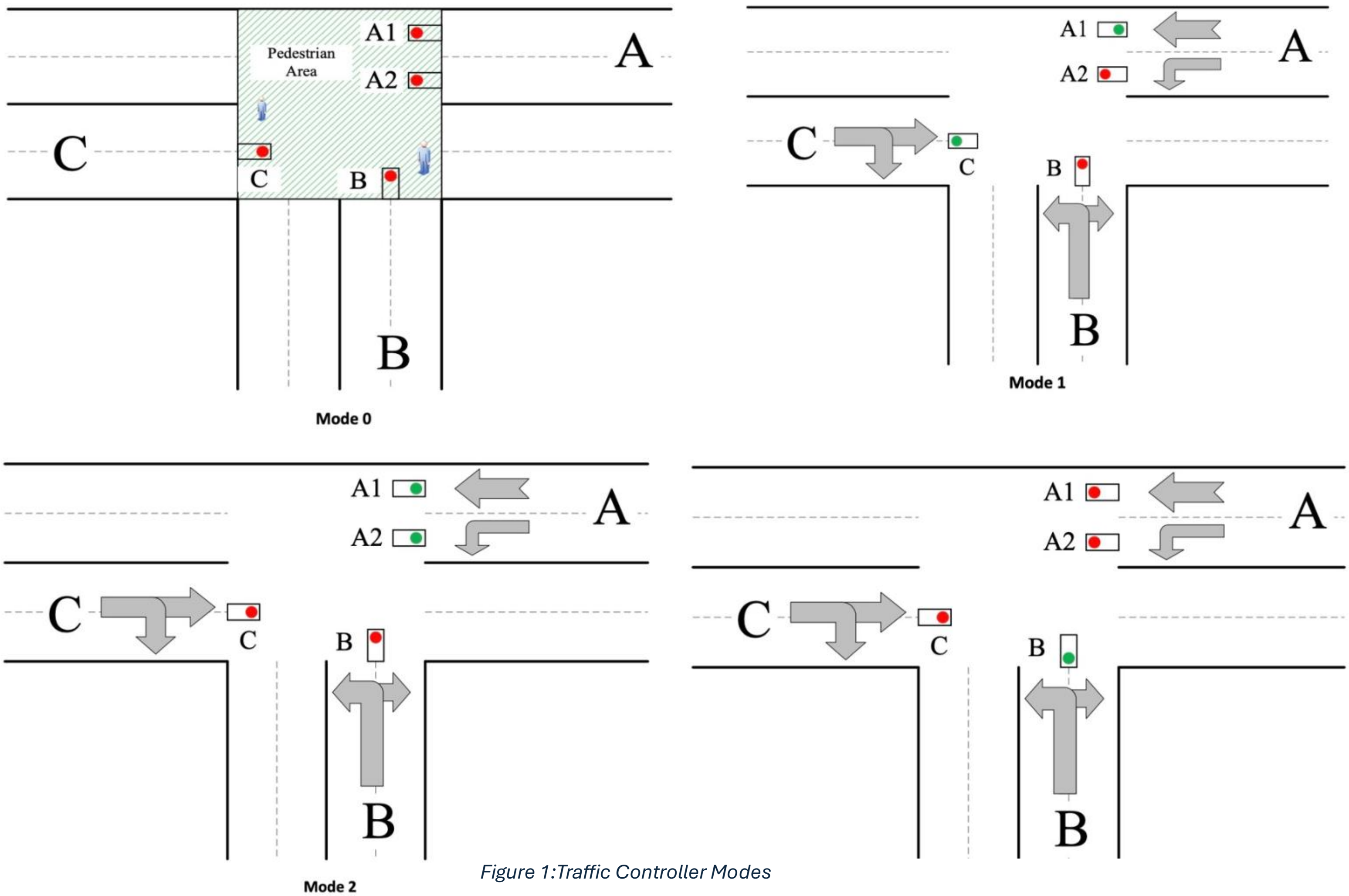

Figure 1:Traffic Controller Modes

As shown in the figure above, the sequence of the controller is:

mode 1 → mode 2 → mode 3 → mode 1→ mode 2 →… etc.

Until the pedestrian button (I) is pressed, the controller tansitions to mode 0 after completing the current mode, which stops all traffic to allow pedestrians to cross the road.

After the period of mode 0 finishes, the controller resumes with the original sequence.

Each mode has specific period as the following:

Table 1: Period of Modes

Mode	**Period (seconds)**
Mode 0	30
Mode 1	30
Mode 2	10
Mode 3	20

3- States

In this section of the report we will explain the states of each mode in this traffic light controller using state table and diagram.

3.1 State Table:

Table 2: State Table

Present State	Pedestrian Button (I)	Next State	Period (seconds)	A1 A2 B C
0 0	0	0 1	30	0 0 0 0
	1	0 1	30	0 0 0 0
0 1	0	1 0	30	1 0 0 1
	1	0 0	0	0 0 0 0
1 0	0	1 1	10	1 1 0 0
	1	0 0	0	0 0 0 0
1 1	0	0 1	20	0 0 1 0
	1	0 0	0	0 0 0 0

3.2 State Diagram:

**Note:
Time Elapsed t = 1
Time Running t = 0

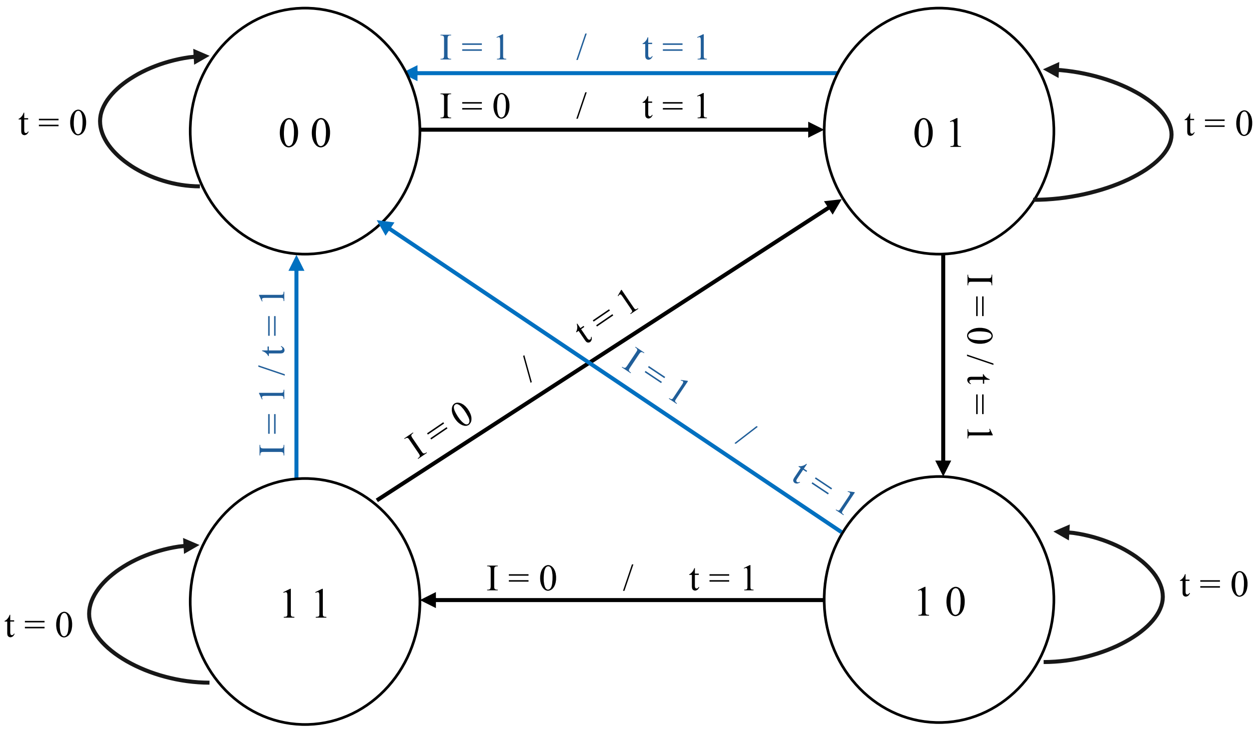

Figure 2: State Diagram

4- Verilog Implementation

4.1 Traffic Controller Verilog Module:

* We built the counter behaviorally not structurally.

```
module t_controller(A1,A2,B,C,mode,clk,I);
  input clk;
  input I;
  output reg [1:0] mode;
  output reg A1, A2, B, C;
```

```
reg [5:0] counter;

always @ (posedge clk)
begin
if (I == 1) begin
    mode <= 2'b00;
    counter <= 0;
  end begin
    case (mode)
      2'b00: if (counter >= 30) begin
               mode <= 2'b01;
               counter <= 0;
             end else begin
               counter <= counter + 1;
             end
      2'b01: if (counter >= 30) begin
               mode <= 2'b10;
               counter <= 0;
              end else begin
               counter <= counter + 1;
              end
      2'b10: if (counter >= 10) begin
               mode <= 2'b11;
               counter <= 0;
              end else begin
               counter <= counter + 1;
              end
      2'b11: if (counter >= 20) begin
               mode <= 2'b01;
               counter <= 0;
              end else begin
               counter <= counter + 1;
              end
      default: begin
                 mode <= 2'b01;
                 counter <= 0;
               end
    endcase
  end
```

```
    end

    always @(*)
    begin
    case  (mode)
        2'b00: {A1, A2, B, C} = 4'b0000;
        2'b01: {A1, A2, B, C} = 4'b1001;
        2'b10: {A1, A2, B, C} = 4'b1100;
       2'b11: {A1, A2, B, C} = 4'b0010;
        default: {A1, A2, B, C} = 4'b0000;
      endcase
    end
endmodule
```

4.2 TestBenches:

4.2.1 Pedestrian Button Not Pressed:

* We used $monitor intsead of $display to prevent using loops.

```
module t_controller_tb;
  reg tclk;
  reg tI;
  wire [1:0] tmode;
  wire tA1, tA2, tB, tC;

  t_controller t1 (tA1,tA2,tB,tC,tmode,tclk,tI);

    initial begin
    tclk = 1;
    tI = 0;
    #1100 $finish;
    end
  always #5 tclk = ~tclk;

  initial
  begin
  $monitor("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
  end
  endmodule
```

```
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
main.v:71: $finish called at 1100 (1s)
```

Figure 3: Pedestrian Not Pressed Output

4.2.2 Pedestrian Button Pressed On Once:

```
module t_controller_tb;
  reg tclk;
  reg tI;
  wire [1:0] tmode;
  wire tA1, tA2, tB, tC;

  t_controller t1 (tA1,tA2,tB,tC,tmode,tclk,tI);

  always #5 tclk = ~tclk;

  initial
  begin
  tclk=1; tI=1;
 #300 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
 #300 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
 #100 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
 #200 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
```

```
 #300 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
 #100 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
 #200 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 #300 $finish;
  end
  endmodule
```

Terminal

```
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
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:86: $finish called at 1800 (1s)
```

Figure 4: Pedestrian ON Once Output

4.2.3 Pedestrian Button Pressed On Twice:

```
module t_controller_tb;
  reg tclk;
  reg tI;
  wire [1:0] tmode;
  wire tA1, tA2, tB, tC;

  t_controller t1 (tA1,tA2,tB,tC,tmode,tclk,tI);

  always #5 tclk = ~tclk;

  initial
```

```
 begin
  tclk=1; tI=1;
 #300 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
 #300 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
 #100 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=1;
 #200 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
 #300 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
 #100 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 tclk=1; tI=0;
 #200 $display("Mode %d: I= %b , A1= %b, A2= %b, B= %b, C=
%b",tmode,tI,tA1,tA2,tB,tC);
 #300 $finish;
  end
  endmodule
```

Terminal

```
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 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:86: $finish called at 1800 (1s)
```

Figure 5: Pedestrian ON Twice Output

5-Conclusion

In this project, we have designed and implemented a traffic signal controller for a T-intersection using Verilog. The controller is able to efficiently handle vehicular movements and enhance pedestrian safety, following a sequence of traffic modes and allowing pedestrians to safely cross the intersection. This project demonstrates the effectiveness of using Verilog to design and implement a traffic signal controller, and it highlights the potential of such controllers to improve intersection management systems.