ELL-201 Project Report
Traffic light controller with delay control

Prateek Mourya - 2022MT11937

April 27, 2024

1 Introduction

Traffic lights play the most important role in our lives while dealing with huge traffic and an increasing population. This controller system is used to design the Traffic lights with the concept of 4-way roads system.

Traffic lights at different locations have different delay times between when a plane gets a green light. In fact some intersections have different amounts of time set for different lanes (such as an intersection between a highway and a street)

2 Implementation

The implementation is on a CPLD board using Verilog. There are 6 inputs mapped to six switched on the CPLD board. There are also 8 outputs mapped to the 8 leds on the board.
4 internal registers d[3:0] of biy-length 8 stores an integer that denotes the amount of time each lane is green. Another register c of length 8 serves as a counter. Every clock cycle the counter c is incremented by 1. Another variable grn stores the current state of the traffic lights with grn = 0,1,2 and 3 denotes the states where lane 1,2,3 and 4 are green respectively. As long as the counter c is less max count d[grn] the state does not change. When the counter reaches the max count d[grn] the state is change by incrementing the variable grn modulo 4 (to keep the value between 0 and 3). The counter c is also reset to 0. Finally the outputs r[3:0] show which lanes have a red light and g[3:0] show which lanes have a green light.

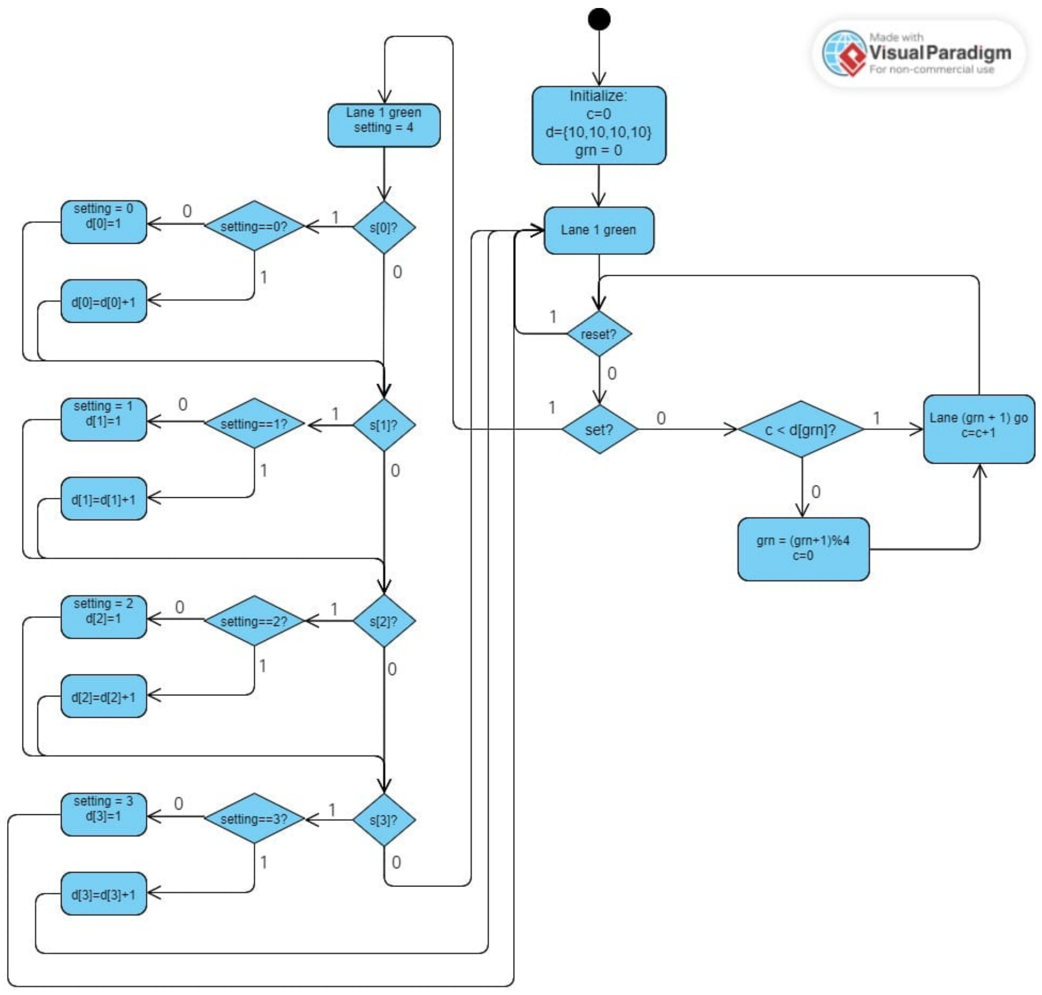
$$\begin{tikzcd}
1110 \arrow[r, "c = d[0]"'] \arrow[loop above, "c < d[0]"] & 1101 \arrow[d, "c = d[1]"] \arrow[loop right, "c < d[1]"] \\
0111 \arrow[u, "c = d[3]"] \arrow[loop below, "c < d[3]"] & 1011 \arrow[l, "c = d[2]"'] \arrow[loop below, "c < d[2]"]
\end{tikzcd}$$

A simplified state diagram for the red lights is shown below.

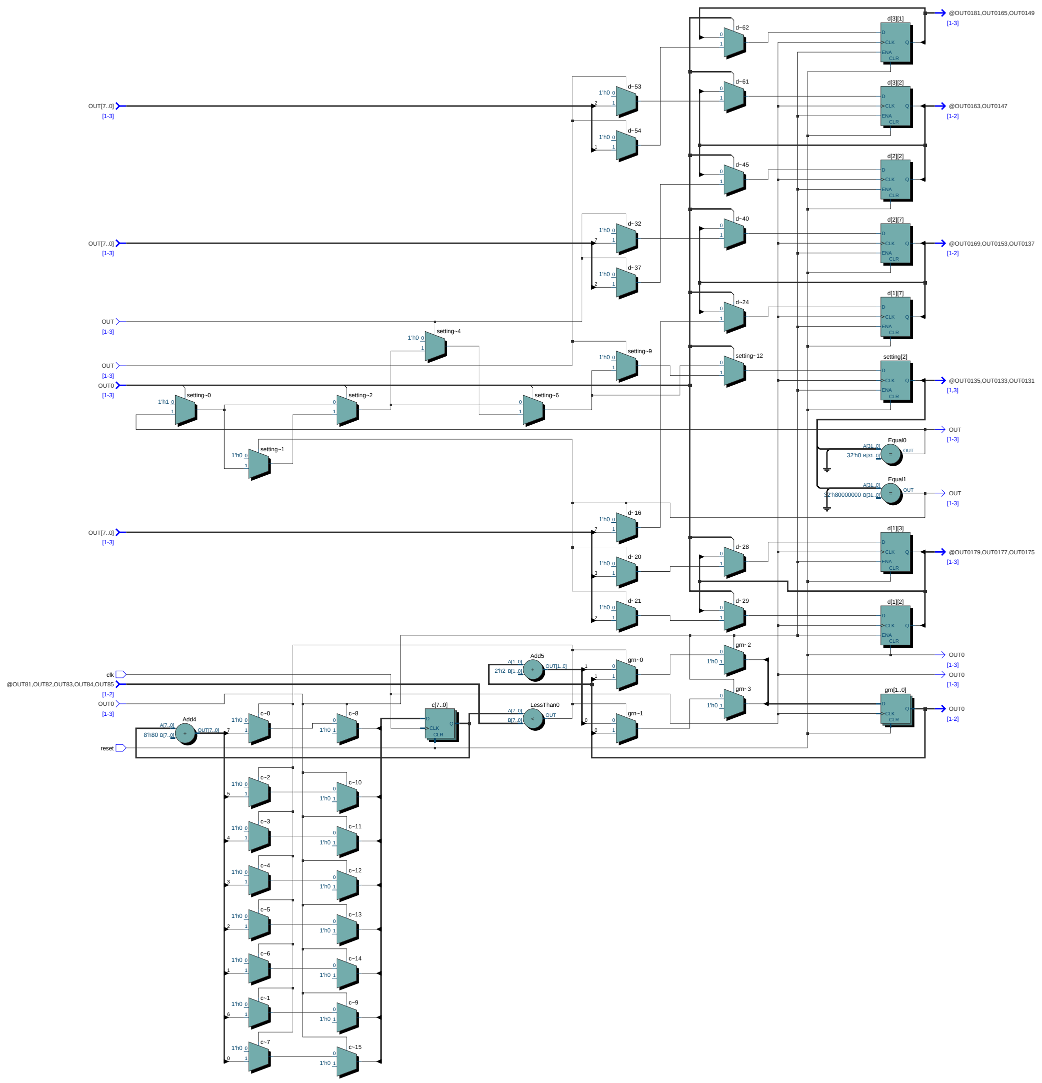
$$\begin{tikzcd}
0001 \arrow[r, "c = d[0]"'] \arrow[loop above, "c < d[0]"] & 0010 \arrow[d, "c = d[1]"] \arrow[loop right, "c < d[1]"] \\
1000 \arrow[u, "c = d[3]"] \arrow[loop below, "c < d[3]"] & 0100 \arrow[l, "c = d[2]"'] \arrow[loop below, "c < d[2]"]
\end{tikzcd}$$

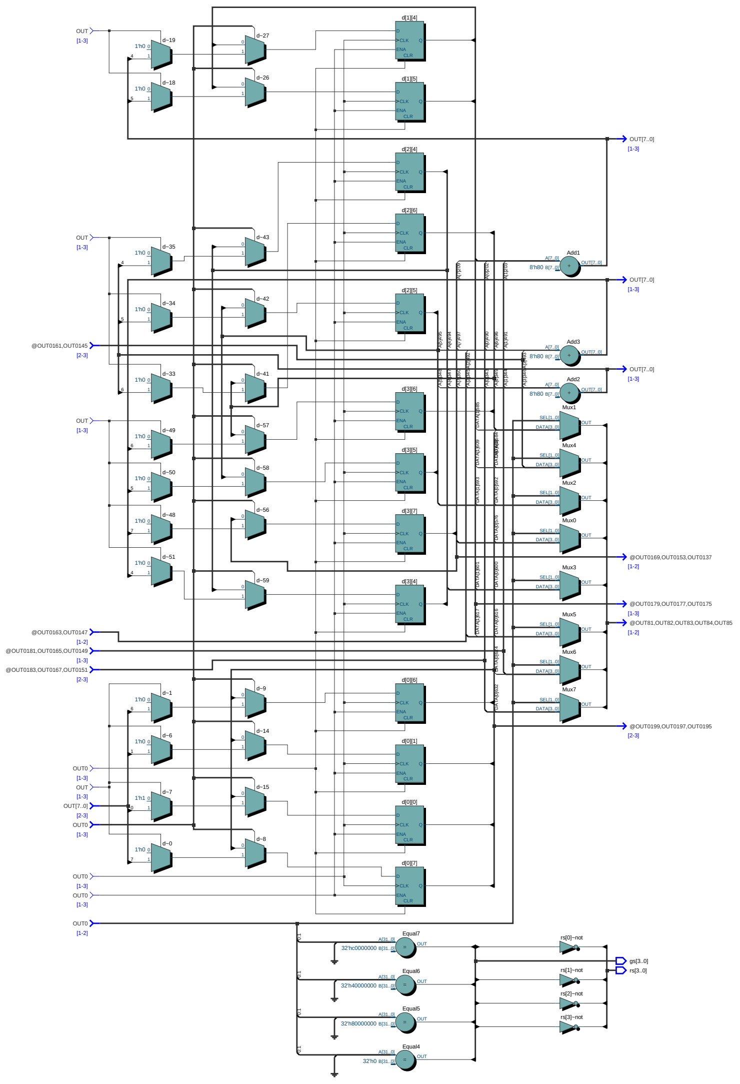
A simplified state diagram for the green lights is shown below.

3 Finite state machine Diagram

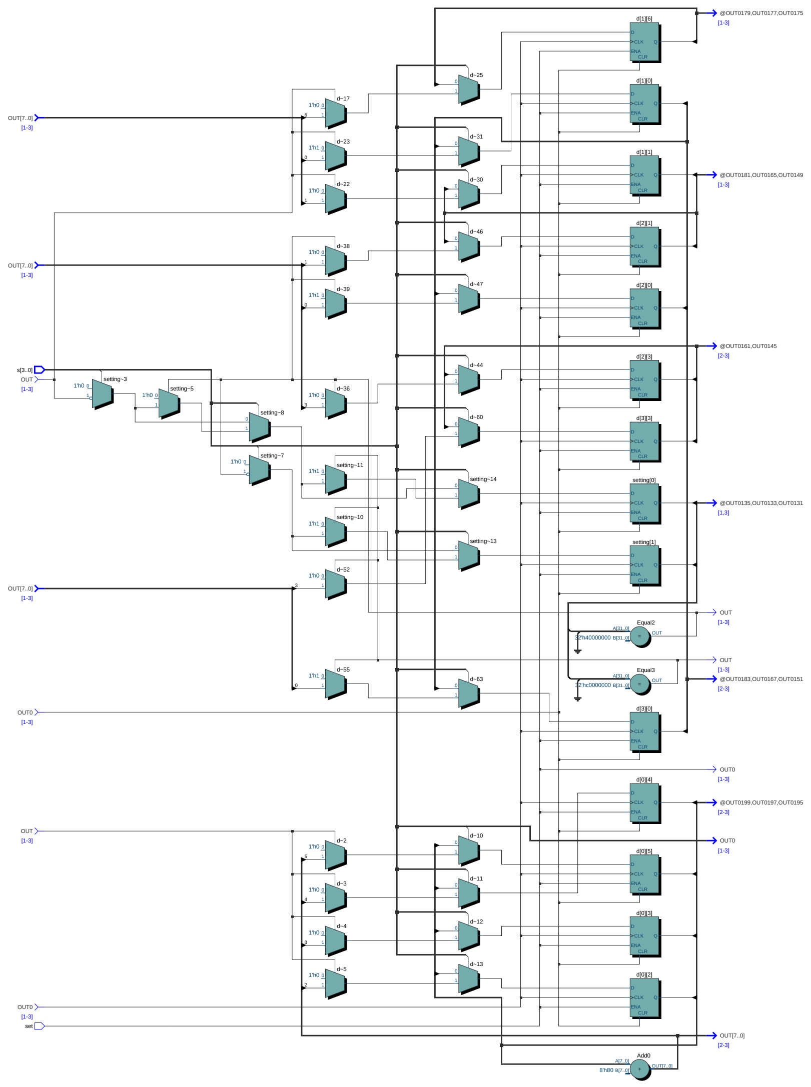


4. Circuit diagram - generated by Quartus





OUT
[1-3]
d~19
d~27
d~18
d~26
d[1][4]
d[1][5]
OUT[7..0]
d[2][4]
d[2][6]
d~35
d~43
Add1
d~34
d~42
d[2][5]
@OUT0161,OUT0145
[2-3]
Add3
Add2
d~33
d~41
d[3][6]
d~49
d~57
Mux1
Mux4
d~50
d~58
d[3][5]
Mux2
d~48
d~56
d[3][7]
Mux0
d~51
@OUT0169,OUT0153,OUT0137
[1-2]
d~59
d[3][4]
Mux3
@OUT0179,OUT0177,OUT0175
Mux5
@OUT81,OUT82,OUT83,OUT84,OUT85
@OUT0163,OUT0147
@OUT0181,OUT0165,OUT0149
@OUT0183,OUT0167,OUT0151
Mux6
Mux7
d~1
d~9
d[0][6]
@OUT0199,OUT0197,OUT0195
d~6
d~14
d[0][1]
OUT0
d~7
d~15
d[0][0]
d~0
d~8
d[0][7]
Equal7
32'hc0000000
rs[0]~not
gs[3..0]
rs[3..0]
Equal6
32'h40000000
rs[1]~not
rs[2]~not
Equal5
32'h80000000
rs[3]~not
Equal4
32'h0



OUT[7..0]
[1-3]
s[3..0]
OUT
set
OUT0
setting~3
setting~5
setting~8
setting~7
setting~11
setting~10
setting~14
setting~13
d~17
d~23
d~22
d~38
d~39
d~36
d~52
d~55
d~2
d~3
d~4
d~5
d~25
d~31
d~30
d~46
d~47
d~44
d~60
d~63
d~10
d~11
d~12
d~13
d[1][6]
d[1][0]
d[1][1]
d[2][1]
d[2][0]
d[2][3]
d[3][3]
setting[0]
setting[1]
d[3][0]
d[0][4]
d[0][5]
d[0][3]
d[0][2]
Equal2
Equal3
Add0
32'h40000000
32'hc0000000
8'h80
@OUT0179,OUT0177,OUT0175
@OUT0181,OUT0165,OUT0149
@OUT0161,OUT0145
@OUT0135,OUT0133,OUT0131
@OUT0183,OUT0167,OUT0151
@OUT0199,OUT0197,OUT0195
[2-3]

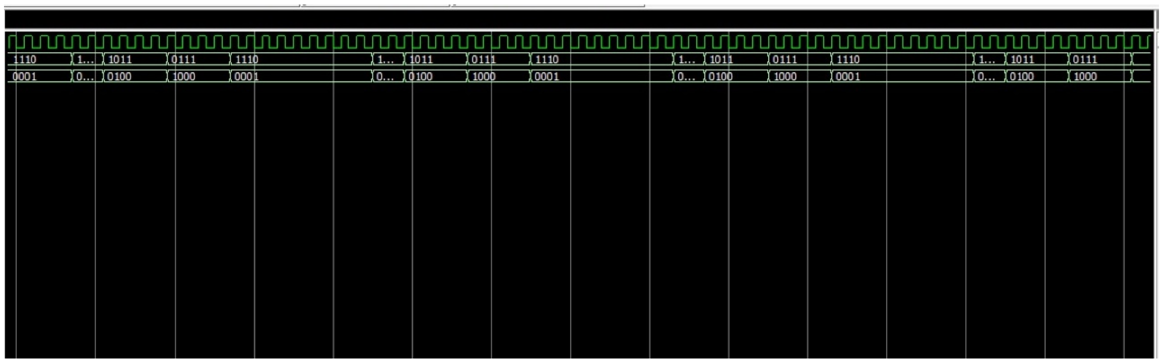
5 Verilog Code

```verilog
module tlc_proj(
    output[3:0] rs,
    output[3:0] gs,
    input reset,
    input set,
    input[3:0] s,
    input clk
    );
        reg[7:0] d[3:0];
        reg[7:0] c = 0;
        reg[1:0] grn = 0;
        reg[2:0] setting = 0;
        integer i;
        reg[3:0] r;
        reg[3:0] g;
        assign rs = r;
        assign gs = g;
        initial
        begin
        d[0]=0;
        d[1]=0;
        d[2]=0;
        d[3]=0;
        end
    always @(posedge clk or posedge reset or posedge set or posedge s[0] or
    posedge s[1] or posedge s[2] or posedge s[3])
    begin
    if(set)
        begin
        setting=4;
        for(i=0;i<4;i=i+1)
        begin
        if(s[i])
        begin
        if(setting==i)
        begin
        d[i] = d[i]+1;
        end
        else
        begin
        setting = i;
        d[i] = 1;
        end
        end
        end
        c = 0;
        grn=0;
        end
        else if(reset)
        begin
        c = 0;
        grn = 0;
        end
        else
        begin
        if(c<d[grn])
        begin
        c=c+1;
        end
        else
        begin
        c=0;
        grn = (grn+1)%4;
        end
        end
        r[0]=1;
        r[1]=1;
        r[2]=1;
        r[3]=1;
        g[0]=0;
        g[1]=0;
        g[2]=0;
        g[3]=0;
        r[grn]=0;
        g[grn]=1;
        end
    endmodule
```

5.1 Testbench:

```
`timescale 1ms / 1us

module testbench_ryan();

  // Inputs
  reg clk;
  reg reset;
  reg set;
reg [3:0] timer;

  // Outputs
  wire [3:0] red;
  wire [3:0] green;

  // Instantiate the module to be tested
  tlc_proj UUT(
      .clk(clk),
      .reset(reset),
      .set(set),
      .rs(red),
      .gs(green),
.s(timer)
  );

  // Clock generation
  always #1 clk = ~clk; // Toggle clock every 1 time unit

  // Initial reset assertion
  initial begin
    clk = 0;
    #1 reset = 1; // Set reset to active (low) state
    set = 0;
    timer = 4'b0000;
    #4 reset = 0; // De-assert reset after 5 time units
set = 1;
timer[0]=1;
# 15 timer[0]=0;
timer[1]=1;
# 1 timer[1]=0;
timer[2]=1;
# 5 timer[2]=0;
timer[3]=1;
# 4 timer[3]=0;
set = 0;
  end

  integer f;
  // Monitor and write to file
  initial begin
    // Open file for writing
    f = $fopen("output_ryan.txt", "w");

    // Monitor changes in number_o and write to file
    repeat (100) begin
      @(posedge clk or posedge reset); // Wait for the next positive clock edge
    case(green)
    4'b0001: $fwrite(f,"Lane 1 Go\n");
    4'b0010: $fwrite(f,"Lane 2 Go\n");
    4'b0100: $fwrite(f,"Lane 3 Go\n");
    4'b1000: $fwrite(f,"Lane 4 Go\n");
    endcase
      //$fwrite(f, "%d %d\n", red,green);
    end

    // Close the file
    $fclose(f);

    // Finish simulation after writing to file
    $finish;
  end

endmodule
```

6 Test Bench Waveforms



The waveform above shows the clock clk. The numbers below show the value of r and g, respectively. So 1101 and 0010 denote red light in lane 1,3, and 4 and green light in lane 2. (lane 1 is denoted by the right most bit). Note the varied time which the 4 lanes stay green. This is due to values set for d using the inputs s[3:0] in the testbench.

7 Applications

The Automatic Traffic Light Controller in the Railway System is positioned at a Four-Line Central-Based Server. It efficiently manages railway traffic using an automatic design system. This controller also oversees road services by regulating traffic flow through the application of digital electronics. Additionally, it allows for the incorporation of delays for the four lanes based on the traffic density.