

Jonathan Lin, Khondakar Mujtaba

ELEN 603

03/23/2025

# Table of Contents

[Table of Contents 2](#_uedi1g95c5p2)

[Introduction 3](#_fj13nz7xyzpv)

[Background 3](#_kd4bbm45oqlj)

[Design Overview 3](#_jlu2t9squmt)

[General Implementation 4](#_u507ryuaxelo)

[Flexible Phase Timing 6](#_ynvvn6lcepyt)

[Leading Pedestrian Intervals 6](#_o5cek3m4jps0)

[Transit Priority 7](#_ynwiclk4vqev)

[Synthesized Design 7](#_muk2xzygcsh1)

[Simulation Results 10](#_fu3qgdliza39)

[Conclusion 12](#_oh4orgsf9mfg)

[Appendix 13](#_hqjoi9ux23s3)

[Works Cited 23](#_m34kpgeakgn2)

# 

# Introduction

Traffic congestion is one of the most significant challenges in large cities, negatively impacting other road users as well as the environment. As a whole, automobiles as a means of transportation are deadly and inefficient, causing numerous deaths yearly while decreasing passenger throughput by blocking alternative transit solutions.

Fortunately, modern technologies allow the design of smart traffic systems that can not only optimize traffic flow but also help prevent accidents and protect pedestrians. The following traffic signal control system has been proposed to address these concerns and enhance overall road safety.

# Background

Traditional traffic lights operate on fixed phases, changing signals at predetermined times based on general traffic patterns, such as rush hours. While such a control scheme is cheap and easy to implement, it is also woefully inadequate, leaving cars waiting even if there are no cars on cross streets.

A 2023 IEEE paper authored by Marin Zhilevski, Mikho Mikhov, and Madlena Zhilevska attempts to address the inefficiencies of fixed phase traffic control through the introduction of a more dynamic system. The normal and emergency operation modes of this system are similar to a traditional traffic control system. However, the “traffic jam” mode is where optimizations are made. In the traffic jam mode, “the red light according to the input data from the specialized traffic monitoring devices, can light 62.5 % of the cycle of the first traffic light and 37.5% of the second traffic light cycle, or at a ratio of 75% to 25%.”

This certainly is an improvement on the fixed phase design, sporting better performance in minimizing car stoppage time. However, it still does not address pedestrian safety, nor make any mention about public transit options that also follow traffic lights such as buses, trams, and light rail trains.

# Design Overview

For our design, we seek to improve on the proposed traffic light control system in three ways. First, we implement fully flexible phases which will allow a complete minimization of vehicle stoppage time. Next, we add leading pedestrian intervals to the phases to increase pedestrian safety. Finally, we give public transport options priority to get them through the intersection as fast as possible.

It may seem pointless to allow pedestrians to walk before cars can drive, but it helps increase pedestrian safety in two ways. First it lets pedestrians limit the amount of time they are on the road while cars are moving next to them. Second, it increases their visibility because they would be halfway in the road when the light turns green. Overall, the city of Portland found that implementing leading pedestrian intervals was “shown to reduce pedestrian-vehicle collisions as much as 60 percent at treated intersections, according to the Urban Street Design Guide by the National Association of City Transportation Officials.”

Similarly, it may seem counterintuitive to hold up a line of cars in order to let a bus cross. However, the reality is that the number of people in that line of cars is likely to be equal if not less than the number of people in a bus.

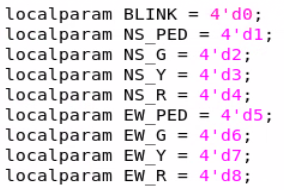


*Figure 1. Picture by Juan de Dios Ortúzar*

So by prioritizing buses and other forms of mass transit, we are actually increasing the throughput of passengers on the road.

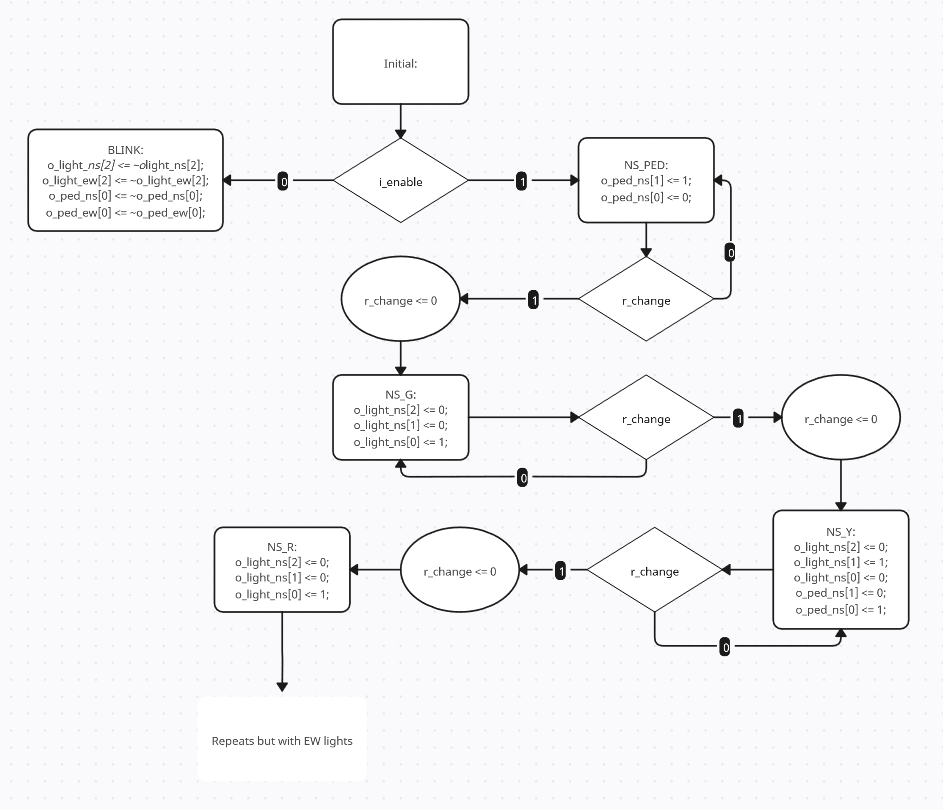
## General Implementation

Our design consists of a state machine with nine different states, each relating to the activation of a light in a certain direction.



*Figure 2. Machine states*

We are assuming that the intersection this traffic system is for consists of one lane for every direction, meaning a two lane wide road. As such, parallel lanes are compatible while perpendicular lanes are not. With this in mind, we coupled together the lights for north and southbound lanes (indicated by “ns”) and east and westbound lanes (indicated by “ew”).



*Figure 3. ASM chart, leaves out EW states*

Since the logic for deciding whether or not to change signals is pretty complex, we opted to utilize a register as a flag to change states for better readability.

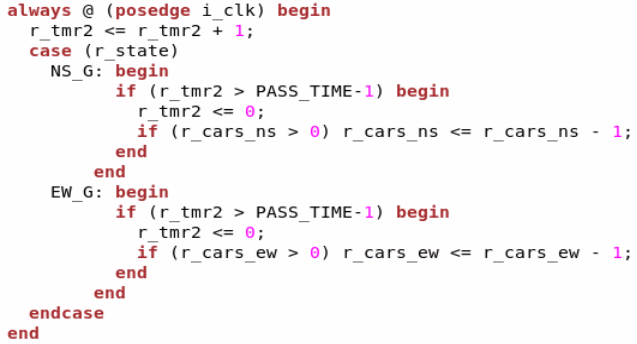
## Flexible Phase Timing

In order to implement flexible phase timing, we utilize a register that holds the number of cars in each paired lane.



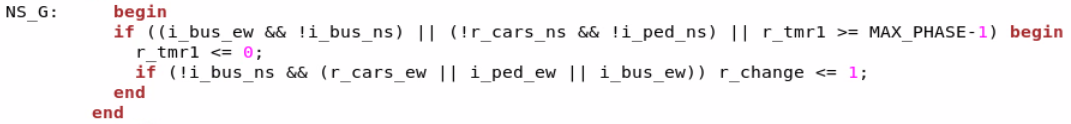
*Figure 4. Loading external data regarding traffic buildup*

We then decrement the number of cars in the lane while the corresponding green light is on, modeling cars driving when the light is green.



*Figure 5. Always block decrementing car counting register*

Finally, in our control logic, we check to see whether or not there are still cars, buses, or pedestrians traveling.

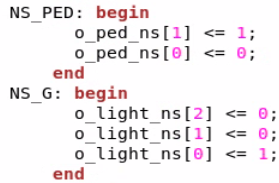


*Figure 6. Control logic while in green light state*

We also have a timer that counts till a maximum phase time in order to prevent a light from being green infinitely.

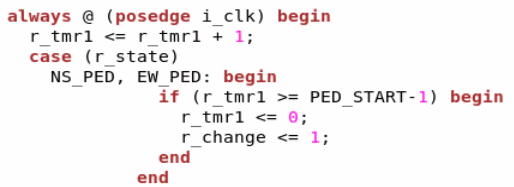
## Leading Pedestrian Intervals

To implement leading pedestrian intervals, we simply include a state that turns on the walk signals before the state that turns the light green.



*Figure 7. Pedestrian light state ahead of car light state*

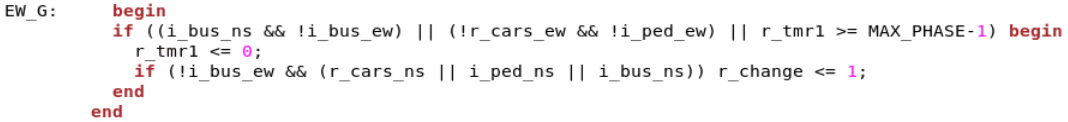
We then utilize a timer to delay a set number of cycles before heading to the next state.



*Figure 8. Always block counts cycles for pedestrian headstart*

## Transit Priority

Finally, to integrate transit priority, we simply check for buses in the control logic.



*Figure 9. Control logic while in green light state*

While the light is green, if there is no bus in the current lane but there is a bus in the perpendicular lane, then switch immediately. However, is there is a bus in the current lane, stay until the bus is gone.

# Synthesized Design

We designed two versions of our Smart Traffic Controller, a parallel implementation and a pipelined Implementation and they take in 8 inputs. There is a clock input i\_clk, an i\_enable input which determines if the traffic light controller is active, i\_ped\_ns and i\_ped\_ew which are signals regarding the direction in which pedestrians want to go, i\_bus\_ns ad i\_bus\_ew which are signals regarding the direction an incoming bus wants to take, i\_cars\_ns and i\_cars\_ew which are 8 bit registers which hold how many cars wish to travel in a certain direction, and finally i\_load which is an input in charge of signaling to the module to load in data regarding cars and pedestrians.

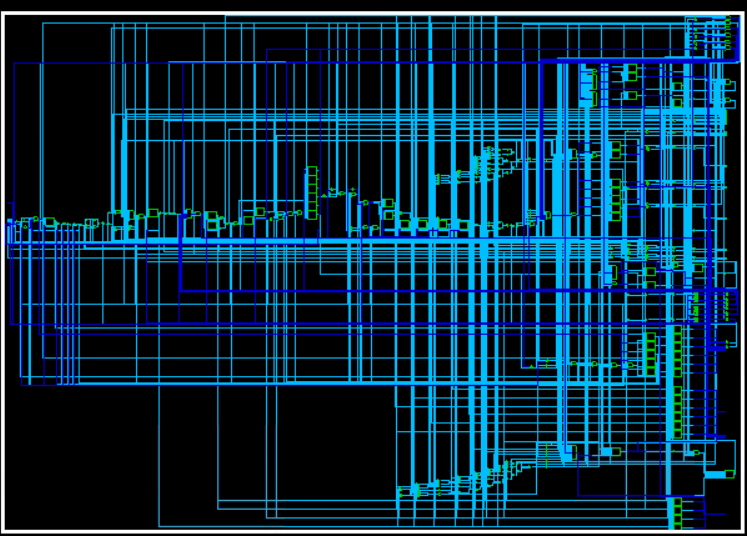
The general structure of our module is that we have one always block that houses the state machine, one always block that handles decrementing the car counter, and one always block that handles light signals. All always blocks are triggered on the positive edge of the clock signal.

The always block in charge of decrementing the car counter has 3 important registers. It has r\_cars\_ns (how many cars want to go north or south) and r\_cars\_ew (how many cars want to go east or west), and it has r\_tmr2 which is used to determine whether or not cars have passed. There is an if-statement in charge of loading in i\_cars\_ns and i\_cars\_ew into r\_cars\_ns and r\_cars\_ew as well as reset r\_tmr2 to 0. Following the if statement is an else statement. It starts with non\_blocking assignments that output r\_cars\_ns and r\_cars\_ew to outputs o\_cars\_ns and o\_cars\_ew for debugging purposes. It is then followed by a case statement which triggers during the greenlight state of either the north-south direction or the east-west direction. In both states is an if statement that will trigger when r\_tmr2 passes the pre-programmed amount of time it takes for a car to pass in which it will decrement r\_cars\_ns or r\_cars\_ew accordingly. This will be used in the state machine.

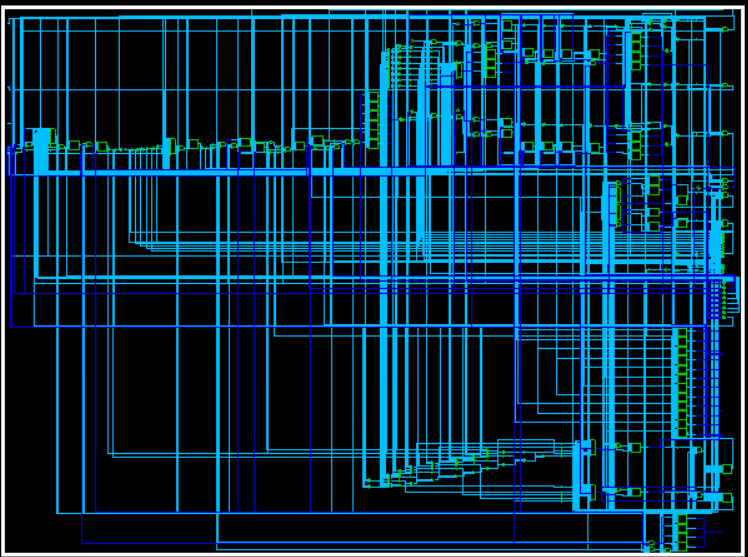
The always block in charge of light signals will act accordingly to the state of the machine. With the exception of the Blink state, all lights are continuous until the end of their state.

When it comes to the state machine, we have 2 different implementations. The parallel implementation has a brute force approach where if a condition for changing the state is reached, it will loop to the next state and reset r\_tmr1 to 0. Otherwise it will increment r\_tmr1. The green signals have an exception where if there is nothing coming in the perpendicular direction, it will just reset r\_tmr1 instead of also switching states. For the pipelined state, we have a similar state machine but it is only used to determine the next state which we put into a buffer called stage2. For green signals when there is no traffic in the perpendicular direction, we set a flag called r\_change which is cleared in the subsequent cycle. We then have a second always block with the job of setting the state of the machine. If either r\_change or i\_load is set we reset r\_tmr. If there is a change of state, we update the state and reset r\_tmr1 as happened for every state in the parallel implementation. Otherwise, we just increment r\_tmr1. And if there is no i\_enable signal, we set the state to blink.

After we simulated our designs, we ran them through Synopsys Design Compiler. We found that our pipelined implementation used up significantly less power and area compared to the parallel implementation. We even added extra logic to skip the pedestrian wait when switching from a red light state and it still took up less area and power. This goes to show that any further modification and improvement of this design should be done so using the pipelined implementation, especially since the only drawback is that there is a 0.5 second delay between state switches.



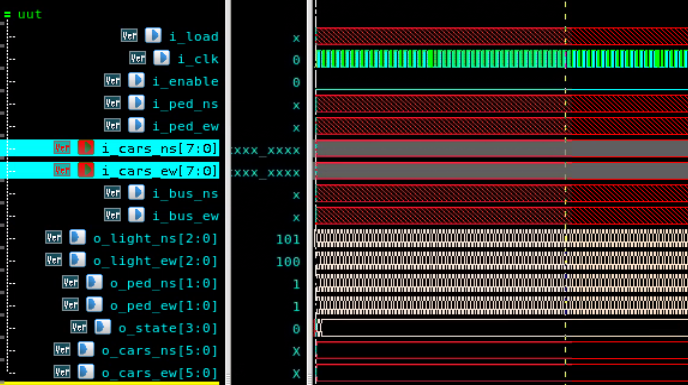
*Figure 10. Parallel Implementation*



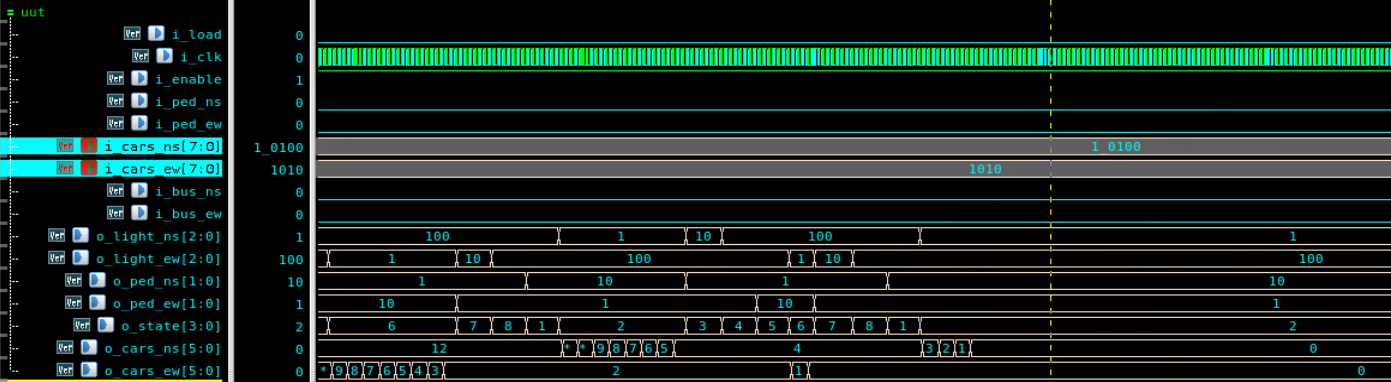
*Figure 11. Pipelined Implementation*

# Simulation Results

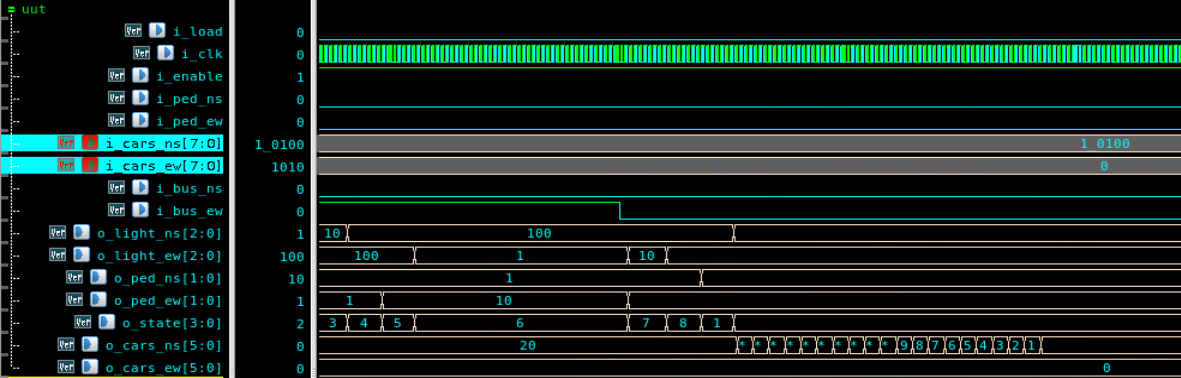
Our testbench first tests the Blink state to make sure that if there is a fault, the traffic lights can still blink red to signal an emergency four way stop. We then test a scenario where 20 cars want to move north or south and 10 cars want to go east or west. We then test a similar scenario except instead of 10 cars moving east or west, we have a bus coming from the east or west. We then have a test scenario of 40 cars that want to move north or south and 10 cars going east or west followed by a similar scenario except a pedestrian also wants to move east or west.



*Figure 12. Blink*



*Figure 13. 20 cars NS, 10 cars EW*



*Figure 14. 20 cars NS, Bus EW*

# Conclusion

Overall, our design improves upon the originally proposed systems using design cues that focus on a less car-centric design. This in turn creates a safer and more efficient intersection. We also found that a pipelined implementation of the design takes up less area and uses less power.

# Appendix

Parallel Implementation

| module Controller(  input i\_load,  input i\_clk,  input i\_enable,  input i\_ped\_ns, i\_ped\_ew,  input [7:0] i\_cars\_ns, i\_cars\_ew,  input i\_bus\_ns, i\_bus\_ew,  output reg [2:0] o\_light\_ns = 0, o\_light\_ew = 0, // 2 = red, 1 = yellow, 0 = green  output reg [1:0] o\_ped\_ns = 0, o\_ped\_ew = 0, // 1 = walk, 0 = don't  output reg [3:0] o\_state,  output reg [5:0] o\_cars\_ns, o\_cars\_ew  );  localparam CPS = 2; // Cycles per second  localparam PED\_START = 5\*CPS; // 5 second ped head start  localparam MAX\_PHASE = 20\*CPS; // 30 second max cycle time  localparam BLINK\_TIME = CPS;  localparam Y\_TIME = 5\*CPS;  localparam R\_TIME = 5\*CPS;  localparam PASS\_TIME = 2\*CPS;  localparam BLINK = 4'd0;  localparam NS\_PED = 4'd1;  localparam NS\_G = 4'd2;  localparam NS\_Y = 4'd3;  localparam NS\_R = 4'd4;  localparam EW\_PED = 4'd5;  localparam EW\_G = 4'd6;  localparam EW\_Y = 4'd7;  localparam EW\_R = 4'd8;  reg [6:0] r\_tmr1 = 0;  reg [6:0] r\_tmr2 = 0;  reg [3:0] r\_state = NS\_G;  reg r\_change = 0;  reg [5:0] r\_cars\_ns, r\_cars\_ew;  always @ (posedge i\_clk) begin  o\_state <= r\_state;  if (i\_load) begin  r\_tmr1 <= 0;  end    else if (i\_enable) begin  case (r\_state)  NS\_PED, EW\_PED: begin  if (r\_tmr1 >= PED\_START-1) begin  r\_tmr1 <= 0;  if (r\_state == 4'd8) r\_state <= NS\_PED;  else r\_state <= r\_state + 1;  end  else begin  r\_tmr1 <= r\_tmr1 + 1;  end  end  NS\_G: begin  if ((i\_bus\_ew && !i\_bus\_ns) || (!r\_cars\_ns && !i\_ped\_ns) || r\_tmr1 >= MAX\_PHASE-1) begin  r\_tmr1 <= 0;  if (!i\_bus\_ns && (r\_cars\_ew || i\_ped\_ew || i\_bus\_ew)) begin  if (r\_state == 4'd8) r\_state <= NS\_PED;  else r\_state <= r\_state + 1;  end;  end  else begin  r\_tmr1 <= r\_tmr1 + 1;  end  end  EW\_G: begin  if ((i\_bus\_ns && !i\_bus\_ew) || (!r\_cars\_ew && !i\_ped\_ew) || r\_tmr1 >= MAX\_PHASE-1) begin  r\_tmr1 <= 0;  if (!i\_bus\_ew && (r\_cars\_ns || i\_ped\_ns || i\_bus\_ns)) begin  if (r\_state == 4'd8) r\_state <= NS\_PED;  else r\_state <= r\_state + 1;  end;  end  else begin  r\_tmr1 <= r\_tmr1 + 1;  end  end  NS\_Y, EW\_Y: begin  if (r\_tmr1 > Y\_TIME-1) begin  r\_tmr1 <= 0;  if (r\_state == 4'd8) r\_state <= NS\_PED;  else r\_state <= r\_state + 1;  end  else begin  r\_tmr1 <= r\_tmr1 + 1;  end  end  NS\_R, EW\_R: begin  if (r\_tmr1 > R\_TIME-1) begin  r\_tmr1 <= 0;  if (r\_state == 4'd8) r\_state <= NS\_PED;  else r\_state <= r\_state + 1;  end  else begin  r\_tmr1 <= r\_tmr1 + 1;  end  end  BLINK: r\_state <= NS\_PED;  endcase  end  else begin  r\_state <= BLINK;  end  end  always @ (posedge i\_clk) begin // decrementing car counter as green light is on.  if (i\_load) begin  r\_tmr2<=0;  r\_cars\_ns <= i\_cars\_ns;  r\_cars\_ew <= i\_cars\_ew;  end  else begin    o\_cars\_ns <= r\_cars\_ns;  o\_cars\_ew <= r\_cars\_ew;  case (r\_state)  NS\_G: begin  if (r\_tmr2 >= PASS\_TIME-1) begin  r\_tmr2 <= 0;  if (r\_cars\_ns > 0) r\_cars\_ns <= r\_cars\_ns - 1;  end  else begin  r\_tmr2 <= r\_tmr2 + 1;  end  end  EW\_G: begin  if (r\_tmr2 >= PASS\_TIME-1) begin  r\_tmr2 <= 0;  if (r\_cars\_ew > 0) r\_cars\_ew <= r\_cars\_ew - 1;  end  else begin  r\_tmr2 <= r\_tmr2 + 1;  end  end    endcase  end  end  always @ (posedge i\_clk) begin  case (r\_state)  BLINK: begin  o\_light\_ns[2] <= ~o\_light\_ns[2];  o\_light\_ew[2] <= ~o\_light\_ew[2];  o\_ped\_ns[0] <= ~o\_ped\_ns[0];  o\_ped\_ew[0] <= ~o\_ped\_ew[0];  end  NS\_PED: begin  o\_ped\_ns[1] <= 1;  o\_ped\_ns[0] <= 0;  end  NS\_G: begin  o\_light\_ns[2] <= 0;  o\_light\_ns[1] <= 0;  o\_light\_ns[0] <= 1;  end  NS\_Y: begin  o\_light\_ns[2] <= 0;  o\_light\_ns[1] <= 1;  o\_light\_ns[0] <= 0;  o\_ped\_ns[1] <= 0;  o\_ped\_ns[0] <= 1;  end  NS\_R: begin  o\_light\_ns[2] <= 1;  o\_light\_ns[1] <= 0;  o\_light\_ns[0] <= 0;  end  EW\_PED: begin  o\_ped\_ew[1] <= 1;  o\_ped\_ew[0] <= 0;  end  EW\_G: begin  o\_light\_ew[2] <= 0;  o\_light\_ew[1] <= 0;  o\_light\_ew[0] <= 1;  end  EW\_Y: begin  o\_light\_ew[2] <= 0;  o\_light\_ew[1] <= 1;  o\_light\_ew[0] <= 0;  o\_ped\_ew[1] <= 0;  o\_ped\_ew[0] <= 1;  end  EW\_R: begin  o\_light\_ew[2] <= 1;  o\_light\_ew[1] <= 0;  o\_light\_ew[0] <= 0;  end  endcase  end  endmodule |
| --- |

Pipelined Implementation

| module Controller(  input i\_load,  input i\_clk,  input i\_enable,  input i\_ped\_ns, i\_ped\_ew,  input [7:0] i\_cars\_ns, i\_cars\_ew,  input i\_bus\_ns, i\_bus\_ew,  output reg [2:0] o\_light\_ns = 0, o\_light\_ew = 0, // 2 = red, 1 = yellow, 0 = green  output reg [1:0] o\_ped\_ns = 0, o\_ped\_ew = 0, // 1 = walk, 0 = don't  output reg [3:0] o\_state,  output reg [5:0] o\_cars\_ns, o\_cars\_ew  );  localparam CPS = 2; // Cycles per second  localparam PED\_START = 5\*CPS; // 5 second ped head start  localparam MAX\_PHASE = 20\*CPS; // 30 second max cycle time  localparam BLINK\_TIME = CPS;  localparam Y\_TIME = 5\*CPS;  localparam R\_TIME = 5\*CPS;  localparam PASS\_TIME = 2\*CPS;  localparam BLINK = 4'd0;  localparam NS\_PED = 4'd1;  localparam NS\_G = 4'd2;  localparam NS\_Y = 4'd3;  localparam NS\_R = 4'd4;  localparam EW\_PED = 4'd5;  localparam EW\_G = 4'd6;  localparam EW\_Y = 4'd7;  localparam EW\_R = 4'd8;  reg [6:0] r\_tmr1 = 0;  reg [6:0] r\_tmr2 = 0;  reg [3:0] stage3 = NS\_G;  reg r\_change = 0;  reg [5:0] r\_cars\_ns, r\_cars\_ew;  reg [3:0] stage2;  always @ (posedge i\_clk) begin //stage 1  if (i\_enable) begin  case (stage3)  BLINK: stage2 <= NS\_PED;    NS\_G: begin  if ((i\_bus\_ew && !i\_bus\_ns) || (!r\_cars\_ns && !i\_ped\_ns) || r\_tmr1 >= MAX\_PHASE-1) begin    if (!i\_bus\_ns && (r\_cars\_ew || i\_ped\_ew || i\_bus\_ew)) begin  stage2 <= NS\_Y;  end  else begin  r\_change <= 1;  end  end  else begin  r\_change<=0;  end  end  EW\_G: begin  if ((i\_bus\_ns && !i\_bus\_ew) || (!r\_cars\_ew && !i\_ped\_ew) || r\_tmr1 >= MAX\_PHASE-1) begin  if (!i\_bus\_ew && (r\_cars\_ns || i\_ped\_ns || i\_bus\_ns)) begin  stage2 <= EW\_Y;  end  else begin  r\_change <= 1;  end  end  else begin  r\_change<=0;  end  end  NS\_Y: begin  if (r\_tmr1 > Y\_TIME-1) begin  stage2 <= NS\_R;  end  end  EW\_Y: begin  if (r\_tmr1 > Y\_TIME-1) begin  stage2 <= EW\_R;  end  end  NS\_R: begin  if (r\_tmr1 > R\_TIME-1) begin  if (i\_ped\_ew) begin  stage2<=EW\_PED;  end  else begin  stage2 <= EW\_G;  end  end  end  EW\_R: begin  if (r\_tmr1 > R\_TIME-1) begin  if (i\_ped\_ns) begin  stage2<=NS\_PED;  end  else begin  stage2 <= NS\_G;  end  end  end  NS\_PED:begin  if (r\_tmr1 >= PED\_START-1) begin  stage2 <= NS\_G;  end  end    EW\_PED: begin  if (r\_tmr1 >= PED\_START-1) begin  stage2 <= EW\_G;  end  end  endcase  end  end  always @ (posedge i\_clk) begin //stage 2    if (i\_enable) begin  if (r\_change | i\_load) begin  r\_tmr1 <= 0;  end  else if (stage2!=stage3) begin  r\_tmr1 <= 0;  o\_state <= stage2;  stage3<=stage2;  end  else begin  r\_tmr1 <= r\_tmr1 + 1;  o\_state <= stage3;  end  end  else begin  stage3 <= BLINK;  end    end  always @ (posedge i\_clk) begin // decrementing car counter as green light is on.  if (i\_load) begin  r\_tmr2<=0;  r\_cars\_ns <= i\_cars\_ns;  r\_cars\_ew <= i\_cars\_ew;  end  else begin    o\_cars\_ns <= r\_cars\_ns;  o\_cars\_ew <= r\_cars\_ew;  case (stage3)  NS\_G: begin  if (r\_tmr2 >= PASS\_TIME-1) begin  r\_tmr2 <= 0;  if (r\_cars\_ns > 0) r\_cars\_ns <= r\_cars\_ns - 1;  end  else begin  r\_tmr2 <= r\_tmr2 + 1;  end  end  EW\_G: begin  if (r\_tmr2 >= PASS\_TIME-1) begin  r\_tmr2 <= 0;  if (r\_cars\_ew > 0) r\_cars\_ew <= r\_cars\_ew - 1;  end  else begin  r\_tmr2 <= r\_tmr2 + 1;  end  end    endcase  end  end  always @ (posedge i\_clk) begin // light management  case (stage3)  BLINK: begin  o\_light\_ns[2] <= ~o\_light\_ns[2];  o\_light\_ew[2] <= ~o\_light\_ew[2];  o\_ped\_ns[0] <= ~o\_ped\_ns[0];  o\_ped\_ew[0] <= ~o\_ped\_ew[0];  end  NS\_PED: begin  o\_ped\_ns[1] <= 1;  o\_ped\_ns[0] <= 0;  end  NS\_G: begin  o\_light\_ns[2] <= 0;  o\_light\_ns[1] <= 0;  o\_light\_ns[0] <= 1;  o\_ped\_ns[1] <= 1;  o\_ped\_ns[0] <= 0;  end  NS\_Y: begin  o\_light\_ns[2] <= 0;  o\_light\_ns[1] <= 1;  o\_light\_ns[0] <= 0;  o\_ped\_ns[1] <= 0;  o\_ped\_ns[0] <= 1;  end  NS\_R: begin  o\_light\_ns[2] <= 1;  o\_light\_ns[1] <= 0;  o\_light\_ns[0] <= 0;  end  EW\_PED: begin  o\_ped\_ew[1] <= 1;  o\_ped\_ew[0] <= 0;  end  EW\_G: begin  o\_light\_ew[2] <= 0;  o\_light\_ew[1] <= 0;  o\_light\_ew[0] <= 1;  o\_ped\_ew[1] <= 1;  o\_ped\_ew[0] <= 0;  end  EW\_Y: begin  o\_light\_ew[2] <= 0;  o\_light\_ew[1] <= 1;  o\_light\_ew[0] <= 0;  o\_ped\_ew[1] <= 0;  o\_ped\_ew[0] <= 1;  end  EW\_R: begin  o\_light\_ew[2] <= 1;  o\_light\_ew[1] <= 0;  o\_light\_ew[0] <= 0;  end  endcase  end  endmodule |
| --- |

Pipelined Implementation power report

| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Report : power  -analysis\_effort low  Design : Controller  Version: V-2023.12-SP3  Date : Sun Mar 23 19:01:32 2025  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Library(s) Used:  gtech (File: /usr/synopsys/syn/V-2023.12-SP3/libraries/syn/gtech.db)  Operating Conditions: ss0p95vn40c Library: saed32rvt\_ss0p95vn40c  Wire Load Model Mode: enclosed  Design Wire Load Model Library  ------------------------------------------------  Controller ForQA saed32rvt\_ss0p95vn40c  \*GEQ\_UNS\_OP\_7\_6\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  \*GT\_UNS\_OP\_7\_4\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  \*GEQ\_UNS\_OP\_7\_4\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  \*NE\_UNS\_OP\_4\_4\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp6\_width4 ForQA saed32rvt\_ss0p95vn40c  \*ADD\_UNS\_OP\_7\_1\_7 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width7 ForQA saed32rvt\_ss0p95vn40c  \*GEQ\_UNS\_OP\_7\_2\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  \*GT\_UNS\_OP\_6\_1\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width6 ForQA saed32rvt\_ss0p95vn40c  \*SUB\_UNS\_OP\_6\_1\_6 ForQA saed32rvt\_ss0p95vn40c  DW01\_dec\_width6 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width6 ForQA saed32rvt\_ss0p95vn40c  DW01\_dec\_width6 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width7 ForQA saed32rvt\_ss0p95vn40c  Global Operating Voltage = 0.95  Power-specific unit information :  Voltage Units = 1V  Capacitance Units = 1.000000ff  Time Units = 1ns  Dynamic Power Units = 1uW (derived from V,C,T units)  Leakage Power Units = 1pW  Attributes  ----------  i - Including register clock pin internal power  Cell Internal Power = 0.0000 uW (0%)  Net Switching Power = 2.5244 uW (100%)  ---------  Total Dynamic Power = 2.5244 uW (100%)  Cell Leakage Power = 0.0000 pW  Information: report\_power power group summary does not include estimated clock tree power. (PWR-789)  Internal Switching Leakage Total  Power Group Power Power Power Power ( % ) Attrs  --------------------------------------------------------------------------------------------------  io\_pad 0.0000 0.0000 0.0000 0.0000 ( 0.00%)  memory 0.0000 0.0000 0.0000 0.0000 ( 0.00%)  black\_box 0.0000 0.0000 0.0000 0.0000 ( 0.00%)  clock\_network 0.0000 0.0000 0.0000 0.0000 ( 0.00%) i  register 0.0000 0.0000 0.0000 0.0000 ( 0.00%)  sequential 0.0000 0.2413 0.0000 0.2413 ( 9.56%)  combinational 0.0000 2.2831 0.0000 2.2831 ( 90.44%)  --------------------------------------------------------------------------------------------------  Total 0.0000 uW 2.5244 uW 0.0000 pW 2.5244 uW |
| --- |

Pipelined Implementation Area Report

| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Report : area  Design : Controller  Version: V-2023.12-SP3  Date : Sun Mar 23 19:01:01 2025  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Information: Changed wire load model for 'DW01\_dec\_width6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*SUB\_UNS\_OP\_6\_1\_6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_dec\_width6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*SUB\_UNS\_OP\_6\_1\_6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_7\_1\_7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_7\_1\_7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_7\_1\_7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GT\_UNS\_OP\_6\_1\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_2\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GT\_UNS\_OP\_6\_1\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_2\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp6\_width4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*NE\_UNS\_OP\_4\_4\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_4\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_4\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GT\_UNS\_OP\_7\_4\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GT\_UNS\_OP\_7\_4\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GT\_UNS\_OP\_7\_4\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GT\_UNS\_OP\_7\_4\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_6\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_6\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Updating design information... (UID-85)  Library(s) Used:  gtech (File: /usr/synopsys/syn/V-2023.12-SP3/libraries/syn/gtech.db)  Number of ports: 558  Number of nets: 1297  Number of cells: 794  Number of combinational cells: 632  Number of sequential cells: 105  Number of macros/black boxes: 0  Number of buf/inv: 93  Number of references: 24  Combinational area: 0.000000  Buf/Inv area: 0.000000  Noncombinational area: 0.000000  Macro/Black Box area: 0.000000  Net Interconnect area: 385.390071  Total cell area: 0.000000  Total area: 385.390071  Information: This design contains unmapped logic. (RPT-7)  Current design is 'Controller'. |
| --- |

Parallel Implementation Power report

| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Report : power  -analysis\_effort low  Design : Controller  Version: V-2023.12-SP3  Date : Sun Mar 23 18:56:13 2025  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Library(s) Used:  gtech (File: /usr/synopsys/syn/V-2023.12-SP3/libraries/syn/gtech.db)  Operating Conditions: ss0p95vn40c Library: saed32rvt\_ss0p95vn40c  Wire Load Model Mode: enclosed  Design Wire Load Model Library  ------------------------------------------------  Controller ForQA saed32rvt\_ss0p95vn40c  \*GEQ\_UNS\_OP\_7\_4\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  \*ADD\_UNS\_OP\_4\_1\_4 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width4 ForQA saed32rvt\_ss0p95vn40c  \*ADD\_UNS\_OP\_7\_1\_7 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width7 ForQA saed32rvt\_ss0p95vn40c  \*GEQ\_UNS\_OP\_7\_6\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width4 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width4 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width7 ForQA saed32rvt\_ss0p95vn40c  \*GT\_UNS\_OP\_7\_4\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width4 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width4 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width7 ForQA saed32rvt\_ss0p95vn40c  \*GEQ\_UNS\_OP\_7\_2\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  \*GT\_UNS\_OP\_6\_1\_1 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width6 ForQA saed32rvt\_ss0p95vn40c  \*SUB\_UNS\_OP\_6\_1\_6 ForQA saed32rvt\_ss0p95vn40c  DW01\_dec\_width6 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width7 ForQA saed32rvt\_ss0p95vn40c  DW01\_cmp2\_width6 ForQA saed32rvt\_ss0p95vn40c  DW01\_dec\_width6 ForQA saed32rvt\_ss0p95vn40c  DW01\_inc\_width7 ForQA saed32rvt\_ss0p95vn40c  \*MULT\_UNS\_OP\_4\_1\_4 ForQA saed32rvt\_ss0p95vn40c  DW02\_mult\_A\_width4\_B\_width1  ForQA saed32rvt\_ss0p95vn40c  DW02\_mult\_A\_width4\_B\_width1  ForQA saed32rvt\_ss0p95vn40c  DW02\_mult\_A\_width4\_B\_width1  ForQA saed32rvt\_ss0p95vn40c  DW02\_mult\_A\_width4\_B\_width1  ForQA saed32rvt\_ss0p95vn40c  DW02\_mult\_A\_width4\_B\_width1  ForQA saed32rvt\_ss0p95vn40c  Global Operating Voltage = 0.95  Power-specific unit information :  Voltage Units = 1V  Capacitance Units = 1.000000ff  Time Units = 1ns  Dynamic Power Units = 1uW (derived from V,C,T units)  Leakage Power Units = 1pW  Attributes  ----------  i - Including register clock pin internal power  Cell Internal Power = 0.0000 uW (0%)  Net Switching Power = 2.7596 uW (100%)  ---------  Total Dynamic Power = 2.7596 uW (100%)  Cell Leakage Power = 0.0000 pW  Information: report\_power power group summary does not include estimated clock tree power. (PWR-789)  Internal Switching Leakage Total  Power Group Power Power Power Power ( % ) Attrs  --------------------------------------------------------------------------------------------------  io\_pad 0.0000 0.0000 0.0000 0.0000 ( 0.00%)  memory 0.0000 0.0000 0.0000 0.0000 ( 0.00%)  black\_box 0.0000 0.0000 0.0000 0.0000 ( 0.00%)  clock\_network 0.0000 0.0000 0.0000 0.0000 ( 0.00%) i  register 0.0000 0.0000 0.0000 0.0000 ( 0.00%)  sequential 0.0000 0.3239 0.0000 0.3239 ( 11.74%)  combinational 0.0000 2.4357 0.0000 2.4357 ( 88.26%)  --------------------------------------------------------------------------------------------------  Total 0.0000 uW 2.7596 uW 0.0000 pW 2.7596 uW |
| --- |

Parallel Implementation area report

| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Report : area  Design : Controller  Version: V-2023.12-SP3  Date : Sun Mar 23 18:55:26 2025  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Information: Changed wire load model for 'DW02\_mult\_A\_width4\_B\_width1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*MULT\_UNS\_OP\_4\_1\_4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW02\_mult\_A\_width4\_B\_width1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*MULT\_UNS\_OP\_4\_1\_4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW02\_mult\_A\_width4\_B\_width1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*MULT\_UNS\_OP\_4\_1\_4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW02\_mult\_A\_width4\_B\_width1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*MULT\_UNS\_OP\_4\_1\_4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW02\_mult\_A\_width4\_B\_width1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*MULT\_UNS\_OP\_4\_1\_4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_4\_1\_4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_4\_1\_4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_dec\_width6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*SUB\_UNS\_OP\_6\_1\_6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_dec\_width6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*SUB\_UNS\_OP\_6\_1\_6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_7\_1\_7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_7\_1\_7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_7\_1\_7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_7\_1\_7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_4\_1\_4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_7\_1\_7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_4\_1\_4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_7\_1\_7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_4\_1\_4' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_inc\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*ADD\_UNS\_OP\_7\_1\_7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GT\_UNS\_OP\_6\_1\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_2\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width6' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GT\_UNS\_OP\_6\_1\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_2\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GT\_UNS\_OP\_7\_4\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GT\_UNS\_OP\_7\_4\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_6\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_6\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for 'DW01\_cmp2\_width7' from '(none)' to 'ForQA'. (OPT-170)  Information: Changed wire load model for '\*GEQ\_UNS\_OP\_7\_4\_1' from '(none)' to 'ForQA'. (OPT-170)  Information: Updating design information... (UID-85)  Library(s) Used:  gtech (File: /usr/synopsys/syn/V-2023.12-SP3/libraries/syn/gtech.db)  Number of ports: 745  Number of nets: 1742  Number of cells: 995  Number of combinational cells: 802  Number of sequential cells: 101  Number of macros/black boxes: 0  Number of buf/inv: 148  Number of references: 25  Combinational area: 0.000000  Buf/Inv area: 0.000000  Noncombinational area: 0.000000  Macro/Black Box area: 0.000000  Net Interconnect area: 419.011685  Total cell area: 0.000000  Total area: 419.011685  Information: This design contains unmapped logic. (RPT-7)    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Report : area  Design : Controller  Version: V-2023.12-SP3  Date : Sun Mar 23 18:55:30 2025  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Library(s) Used:  gtech (File: /usr/synopsys/syn/V-2023.12-SP3/libraries/syn/gtech.db)  Number of ports: 745  Number of nets: 1742  Number of cells: 995  Number of combinational cells: 802  Number of sequential cells: 101  Number of macros/black boxes: 0  Number of buf/inv: 148  Number of references: 25  Combinational area: 0.000000  Buf/Inv area: 0.000000  Noncombinational area: 0.000000  Macro/Black Box area: 0.000000  Net Interconnect area: 419.011685  Total cell area: 0.000000  Total area: 419.011685  Information: This design contains unmapped logic. (RPT-7)  Current design is 'Controller'. |
| --- |

# 

# 

# Works Cited

Ortúzar, Juan de Dios. “Sustainable Urban Mobility: What Can Be Done to Achieve It?” *Journal of the Indian Institute of Science*, vol. 99, no. 4, 28 Nov. 2019, pp. 683–693, https://doi.org/10.1007/s41745-019-00130-y.

“Pedestrian Head Starts | Portland.gov.” *Www.portland.gov*, www.portland.gov/transportation/traffic-operations/pedestrian-head-starts.

Pinder, Matt. “Signals for Change: Ideas from the Dutch for Improving Traffic Lights.” *Beyond the Automobile*, 11 Aug. 2020, beyondtheautomobile.com/2020/08/11/signals-for-change/.

Zhilevski, Marin, et al. *Design of a Traffic Light System Using Verilog HDL*. 8 June 2023, pp. 1–6, https://doi.org/10.1109/hora58378.2023.10156804. Accessed 24 Mar. 2025.