Diane Shan

862148900

EE120A Section 23

Jack Huang

Mini-Project: Traffic Signal

**Purpose**

The purpose of this project is to create a stoplight controller for a corner where two streets cross. There will be two lights and each one is either in a state go or stop. Then the controller will include a crosswalk feature that puts both lights in the stop state for 5 clock cycles.

**Design Constraints**

Some design constraints are that we can’t make it a smart traffic signal where, if there is no one waiting at one light and someone comes to the other light, it will switch to that light to make it faster for the person waiting. Another design constraint is that each light has a set number of cycles it will go through before switching to another state. This can cause inefficiency at a non-busy traffic light. Another design constraint is that there are turn signals, so if someone wanted to turn on the traffic light, then they would not be able to.

**System Design**

*HLSM:*

Diagram

Description automatically generated

*Datapath and Controller:*

Chart, box and whisker chart

Description automatically generated

Diagram

Description automatically generated

*FSM:*

Diagram

Description automatically generated

*Verilog Code:*

***design.sv:***

`timescale 1ns / 1ps

module traffic\_signal

(

input wire B,

input wire ctr,

input wire reset,

output reg ns\_green,

output reg ns\_red,

output reg ew\_green,

output reg ew\_red,

output reg b\_reg,

output reg [2:0]ctr\_reg

);

//declaring states

reg [2:0] current\_state = 3'b000 ;

reg [2:0] next\_state ;

reg n\_reg;

localparam START = 3'b000 ;

localparam TEMP = 3'b001 ;

localparam NS = 3'b010 ;

localparam EW = 3'b011 ;

localparam CROSS = 3'b100 ;

always @ (posedge ctr, posedge reset) begin

if (reset) begin

current\_state <= START;

end

else begin

current\_state <= next\_state;

end

end

always @(\*) begin

next\_state = current\_state;

case(current\_state)

START: begin

n\_reg = 1'b0; //initialize n

ew\_green = 1'b0; //EW green light off

ew\_red = 1'b0; //EW red light off

ns\_green = 1'b0; //NS green light off

ns\_red = 1'b0; //NS red light off

next\_state = TEMP;

end

TEMP: begin

ctr\_reg = 3'b000; //initialize counter to 0

if (n\_reg == 1'b0) begin

n\_reg = 1'b1;

end

else begin

n\_reg = 1'b0;

end

ew\_green = 1'b0; //EW green light off

ew\_red = 1'b1; //EW red light off

ns\_green = 1'b0; //NS green light off

ns\_red = 1'b1; //NS red light off

b\_reg = B;

if (b\_reg == 1'b1) begin

next\_state = CROSS;

end

else if (n\_reg == 1'b1) begin

next\_state = NS;

end

else begin

next\_state = EW;

end

end

NS: begin

ew\_green = 1'b0; //EW green light off

ew\_red = 1'b1; //EW red light on

ns\_green = 1'b1; //NS green light on

ns\_red = 1'b0; //NS red light off

b\_reg = 1'b0;

ctr\_reg = ctr\_reg + 1; //increment ctr

if (ctr\_reg == 3'b011) begin

next\_state = TEMP;

end

else begin

next\_state = NS;

end

end

EW: begin

ew\_green = 1'b1; //EW green light on

ew\_red = 1'b0; //EW red light off

ns\_green = 1'b0; //NS green light off

ns\_red = 1'b1; //NS red light on

b\_reg = 1'b0; //walk not pressed

ctr\_reg = ctr\_reg + 1; //increment ctr

if (ctr\_reg == 3'b011) begin

next\_state = TEMP;

end

else begin

next\_state = EW;

end

end

CROSS: begin

ew\_green = 1'b0; //EW green light off

ew\_red = 1'b1; //EW red light on

ns\_green = 1'b0; //NS green light off

ns\_red = 1'b1; //NS red light on

b\_reg = 1'b1; //walk pressed

ctr\_reg = ctr\_reg + 1; //increment ctr

if (ctr\_reg == 3'b101) begin

b\_reg = 1'b0;

next\_state = TEMP;

end

else begin

next\_state = CROSS;

end

end

endcase

end

endmodule

***testbench.sv***:

`timescale 1ns / 1ps

module traffic\_signal\_tb;

//Inputs

reg B;

reg ctr;

reg reset;

//Outputs

wire ns\_green;

wire ns\_red;

wire ew\_green;

wire ew\_red;

wire b\_reg;

wire [2:0] ctr\_reg;

// Instantiate unit under test

traffic\_signal UUT (

.B(B),

.ctr(ctr),

.reset(reset),

.ns\_green(ns\_green),

.ns\_red(ns\_red),

.ew\_green(ew\_green),

.ew\_red(ew\_red),

.b\_reg(b\_reg),

.ctr\_reg(ctr\_reg)

);

//for clock signals

initial begin

ctr = 1;

#1000 $finish;

end

always #20 ctr = ~ctr;

initial begin

$dumpfile("dump.vcd"); $dumpvars;

$display("Testcase #1");

//assign inputs

B = 1'b1;

reset = 1;

#30

reset = 0;

if (ns\_green == 1'b0 && ns\_red == 1'b1 && ew\_green == 1'b0 && ew\_red == 1'b1)

$display ("\t Testcase #1 successful;");

else

$display("\t Testcase #1 is wrong");

#100

$display("Testcase #2");

B = 1'b0;

if (ns\_green == 1'b0 && ns\_red == 1'b1 && ew\_green == 1'b0 && ew\_red == 1'b1)

$display ("\t Testcase #2 successful");

else

$display("\t Testcase #2 is wrong");

#200

B = 1'b1;

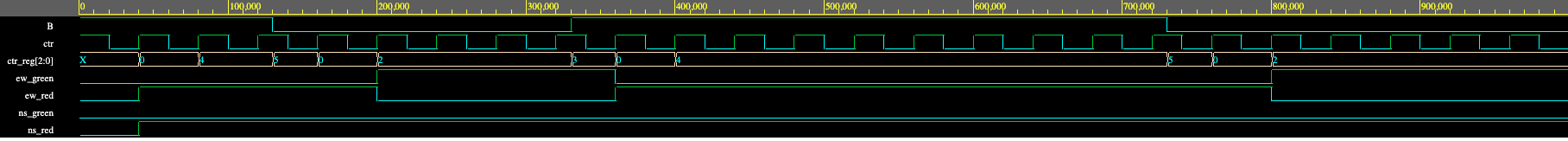
#400

B = 1'b0;

end

endmodule

*Waveform:*



**Problems Encountered**

After coding up the logic from our HLSM, we tried to test the code and found that the state transitions were not working. The state would stay the same the entire time and would not transition to the next state. It would stay at the initial state it is given. This caused us to have to redo our Verilog code and we realized we needed to add another always block. However, after doing that, we realized it would only run until the next state and then stay in that state for the rest of the cycle. As a result, we decided to redo the Verilog code again and were finally able to get the system to change states correctly.

**Conclusion**

The system works according to the provided specifications. The lights change from red to green respectively for each street. Then when the walk sign is pressed, all lights turn red and the walk sign goes for 5 clock cycles. Some ways to improve the system would be to add a yellow light for slow or to add a sensor that would sense whether or not there was someone waiting at a light. If there is, then the light would finish up the clock cycle it is currently on and switch to green for the street that the car is at. Another improvement would be to add turn signals and different crosswalk paths, so when someone is walking, cars going in the same direction can still keep on going.