KONGU ENGINEERING COLLEGE

PERUNDURAI ERODE-638060

ECE DEPARTMENT

VLSI MINI PROJECT

REAL-TIME TRAFFIC LIGHT CONTROL SYSTEM

PRESENTED BY:

1)KARTEESWAR K P 20ECR071

2)KARTHI P 20ECR072

3)KARTHIGA K 20ECR073

OBJECTIVE

- The objective of this project is to develop a traffic light control system at times of emergency conditions or traffic jam conditions using Verilog.
- The algorithm for the implementation is based on Finite State Machine (FSM) .

PROBLEM STATEMENT

- Rising traffic congestion is an inescapable condition in large and growing metropolitan areas across the world.
- Peak-hour traffic congestion is an inherent result of the way modern societies operate. It stems from the widespread desires of people to pursue certain goals that inevitably overload existing roads.
- In spite of attempted remedies, the traffic control system is getting worse and it creates a mess at times of emergency condition.

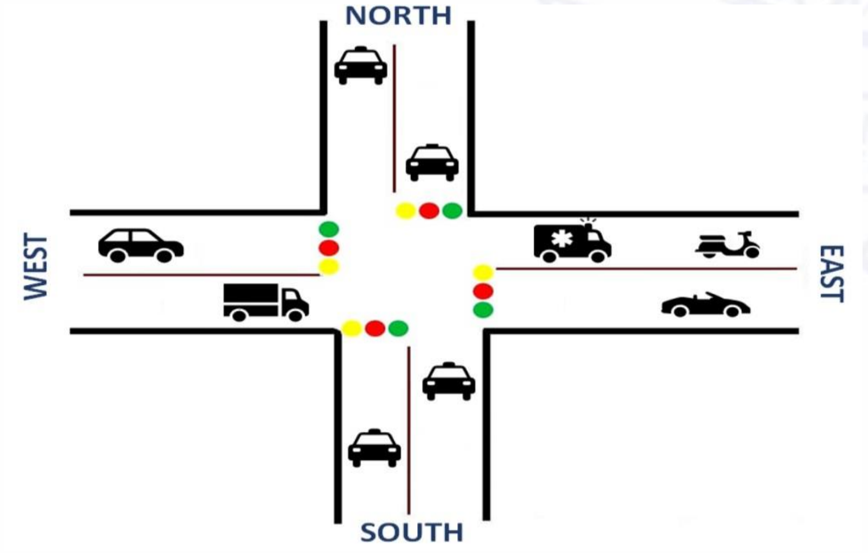
PROBLEM SOLUTION

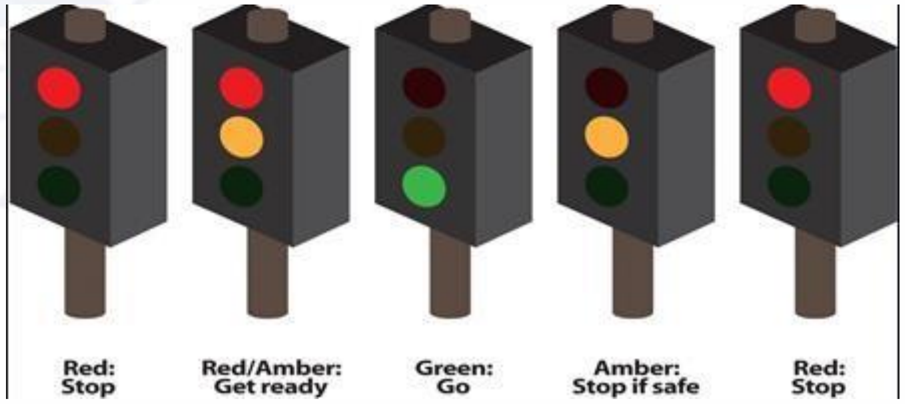
- To encounter the above-mentioned problem an Advanced traffic light controlled system powered with sensors is established.
- Sensors work for the identification of the emergency or traffic jam condition in the specified road and with the input of the sensors, the state switch accordingly.

KONGU ENGINEERING COLLEGE

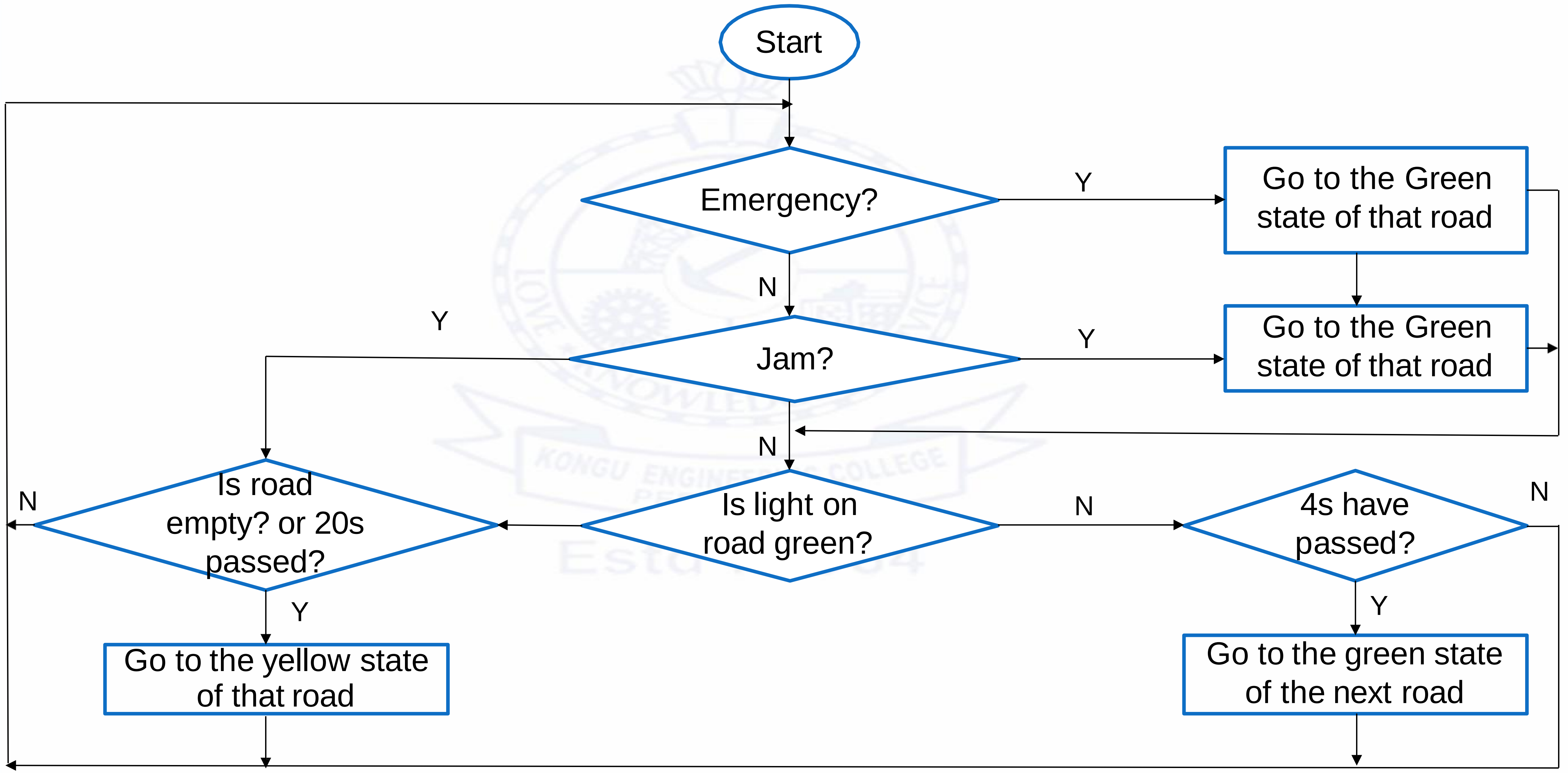
METHODOLOGY

- There are four roads: East, North, West, and South.
- The green light will go on circularly in the counter-clockwise direction.
- The green light will remain on for 20 seconds. In this period all other roads will be red.
- Then, the yellow light will light up and the next road to be green will be in a red-yellow state to caution the drivers.
- The system will be in this state for 4 seconds.

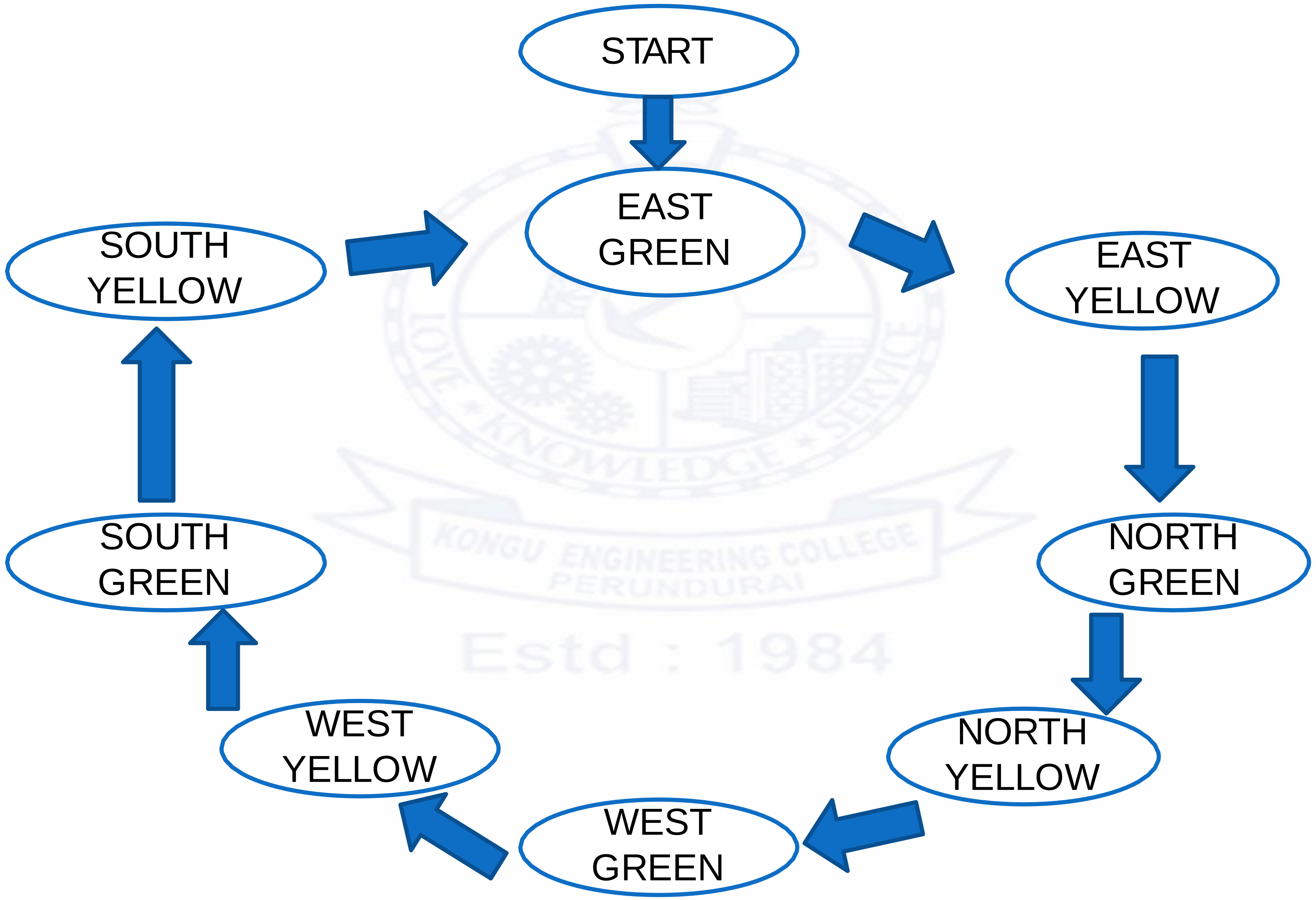




FLOW CHART


Start
Emergency?
Y
Go to the Green state of that road
N
Go to the Green state of that road
Jam?
Y
Y
N
Is light on road green?
N
4s have passed?
N
Is road empty? or 20s passed?
N
Y
Go to the yellow state of that road
Y
Go to the green state of the next road

STATE DIAGRAM



$$\begin{tikzcd}
\text{START} \arrow[d] \\
\text{EAST GREEN} \arrow[r] & \text{EAST YELLOW} \arrow[r] & \text{NORTH GREEN} \arrow[r] & \text{NORTH YELLOW} \arrow[r] & \text{WEST GREEN} \arrow[r] & \text{WEST YELLOW} \arrow[r] & \text{SOUTH GREEN} \arrow[r] & \text{SOUTH YELLOW} \arrow[lllllll, bend left]
\end{tikzcd}$$

DESCRIPTION OF STATES

State	State (Binary)	State Name	East	North	West	South
0	000	East_green	Green	Red	Red	Red
1	001	East_yellow	Yellow	Red-Yellow	Red	Red
2	010	North_green	Red	Green	Red	Red
3	011	North_yellow	Red	Yellow	Red-Yellow	Red
4	100	West_green	Red	Red	Green	Red
5	101	West_yellow	Red	Red	Yellow	Red-Yellow
6	110	South_green	Red	Red	Red	Green
7	111	South_yellow	Red-Yellow	Red	Red	Yellow

EMERGENCY CONDITION

- The emergency condition is implemented by using a 4-bit variable named Emergency.
- If Emergency[3] is 1 then it means that the east road has an emergency condition and the system goes to the east_green state whose binary value is 000.
- Similarly, emergency conditions for other roads are implemented.

DIRECTION	EMERGENCY				STATE		
	Emergency [3]	Emergency [2]	Emergency [1]	Emergency [0]	State[2]	State[1]	State[0]
East	1	0	0	0	0	0	0
North	0	1	0	0	0	1	0
West	0	0	1	0	1	0	0
South	0	0	0	1	1	1	0

TRAFFIC JAM CONDITION

- Similarly, the jam condition is implemented using a 4-bit variable Jam.
- Traffic jam state operation is same as that of the emergency condition.

DIRECTION	JAM				STATE		
	Jam[3]	Jam[2]	Jam[1]	Jam[0]	State[2]	State[1]	State[0]
East	1	0	0	0	0	0	0
North	0	1	0	0	0	1	0
West	0	0	1	0	1	0	0
South	0	0	0	1	1	1	0

VERILOG CODE

```
module traffic_light_controller(East_road,North_road,West_road,South_road,clk,rst,Emergency,Jam,Empty);
input clk,rst;
input [3:0]Emergency,Jam,Empty;//Emergency/Jam/Empty=0000, for east 1000,north 0100,west 0010,south 0001
output reg[2:0]East_road,North_road,West_road,South_road;
//Red=100,Yellow=010,Green=001,Red&Yellow=110,none=000
 reg[2:0] state;
 parameter [2:0] east_green=3'b000;
 parameter [2:0] east_yellow=3'b001;
 parameter [2:0] north_green=3'b010;
 parameter [2:0] north_yellow=3'b011;
 parameter [2:0] west_green=3'b100;
 parameter [2:0] west_yellow=3'b101;
 parameter [2:0] south_green=3'b110;
 parameter [2:0] south_yellow=3'b111;
 reg[4:0] count;
```

VERILOG CODE

```
always@(posedge clk, negedge rst)
  begin
   if(!rst)
    begin
    count=5'b00000;
    end
    else if(|Emergency)
    begin
state={Emergency[1]|Emergency[0],Emergency[2]|
Emergency[0],1'b0};
    count=5'b00000;
    end
   else if(|Jam)
    begin
    state={Jam[1]|Jam[0],Jam[2]|Jam[0],1'b0};
    count=5'b00000;
    end
else
    begin
     case(state)
     east_green:
         begin
          if(count==5'b10011||Empty==4'b1000)
            begin
             count=5'b00000;
             state=east_yellow;
            end
          else
            begin
            count=count+5'b00001;
            state=east_green;
            end
          end
```

VERILOG CODE

```
east_yellow:
        begin
         if(count==5'b00011)
         begin
         count=5'b00000;
         state=north_green;
         end
         else
           begin
           count=count+5'b00001;
           state=east_yellow;
           end
         end
```

```
north_green:
        begin
         if(count==5'b10011||Empty==4'b0100)
         begin
           count=5'b00000;
           state=north_yellow;
           end
         else
           begin
           count=count+5'b00001;
           state=north_green;
           end
         end
```

VERILOG CODE

```
north_yellow:
      begin
       if(count==5'b00011)
         begin
         count=5'b00000;
         state=west_green;
         end
       else
         begin
         count=count+5'b00001;
         state=north_yellow;
         end
       end
```

```
west_green:
  begin
    if(count==5'b10011||Empty==4'b0010)
           begin
            count=5'b00000;
            state=west_yellow;
            end
          else
            begin
            count=count+5'b000001;
            state=west_green;
            end
          end
```

VERILOG CODE

```
west_yellow:
         begin
          if(count==5'b00011)
          begin
          count=5'b00000;
          state=south_green;
          end
          else
            begin
            count=count+5'b00001;
            state=west_yellow;
            end
          end
south_green:
         begin
          if(count==5'b10011||Empty==4'b0001)
           begin
            count=5'b00000;
            state=south_yellow;
            end
          else
            begin
            count=count+5'b00001;
            state=south_green;
            end
          end
south_yellow:
         begin
          if(count==5'b00011)
            begin
            count=5'b00000;
            state=east_green;
            end
          else
```

VERILOG CODE

```
            begin
            count=count+5'b00001;
            state=south_yellow;
            end
            end
default:
        begin
         count=5'b00000;
         state=east_green;
       end
     endcase
   end
  end
//Red=100,Yellow=010,Green=001,Red&Yellow=110,
none=000
 always@(state)
   begin
```

```
if(!rst)
     begin
     East_road=3'b000;
     North_road=3'b000;
     West_road=3'b000;
     South_road=3'b000;
     end
   else
    begin
    case(state)
  east_green:
     begin
      East_road=3'b001;
      North_road=3'b100;
     West_road=3'b100;
     South_road=3'b100;
     end
```

VERILOG CODE

```
east_yellow:
       begin
       East_road=3'b010;
       North_road=3'b110;//
       West_road=3'b100;
       South_road=3'b100;
       end

north_green:
       begin
        East_road=3'b100;
       North_road=3'b001;
       West_road=3'b100;
       South_road=3'b100;
       end
```

```
north_yellow:
       begin
        East_road=3'b100;
        North_road=3'b010;
       West_road=3'b110;//
       South_road=3'b100;
       end

west_green:
       begin
        East_road=3'b100;
       North_road=3'b100;
       West_road=3'b001;
       South_road=3'b100;
       end
```

VERILOG CODE

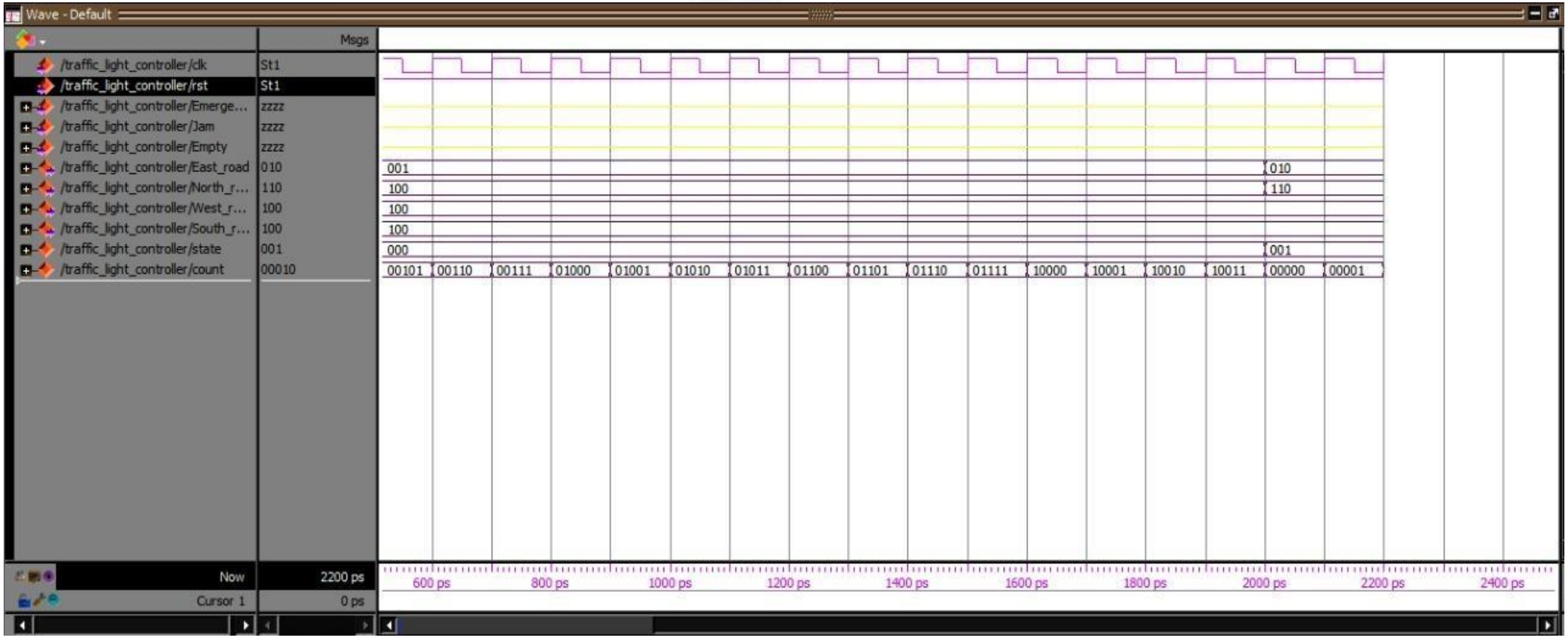
```
west_yellow:
       begin
        East_road=3'b100;
        North_road=3'b100;
       West_road=3'b010;
       South_road=3'b110; //
       end

      south_green:
       begin
        East_road=3'b100;
       North_road=3'b100;
       West_road=3'b100;
       South_road=3'b001;
       end
```

```
south_yellow:
       begin
        East_road=3'b110;//
        North_road=3'b100;
       West_road=3'b100;
       South_road=3'b010;
       end
     endcase
    end
   end
endmodule
```

KONGU ENGINEERING COLLEGE

OUTPUT WAVEFORM



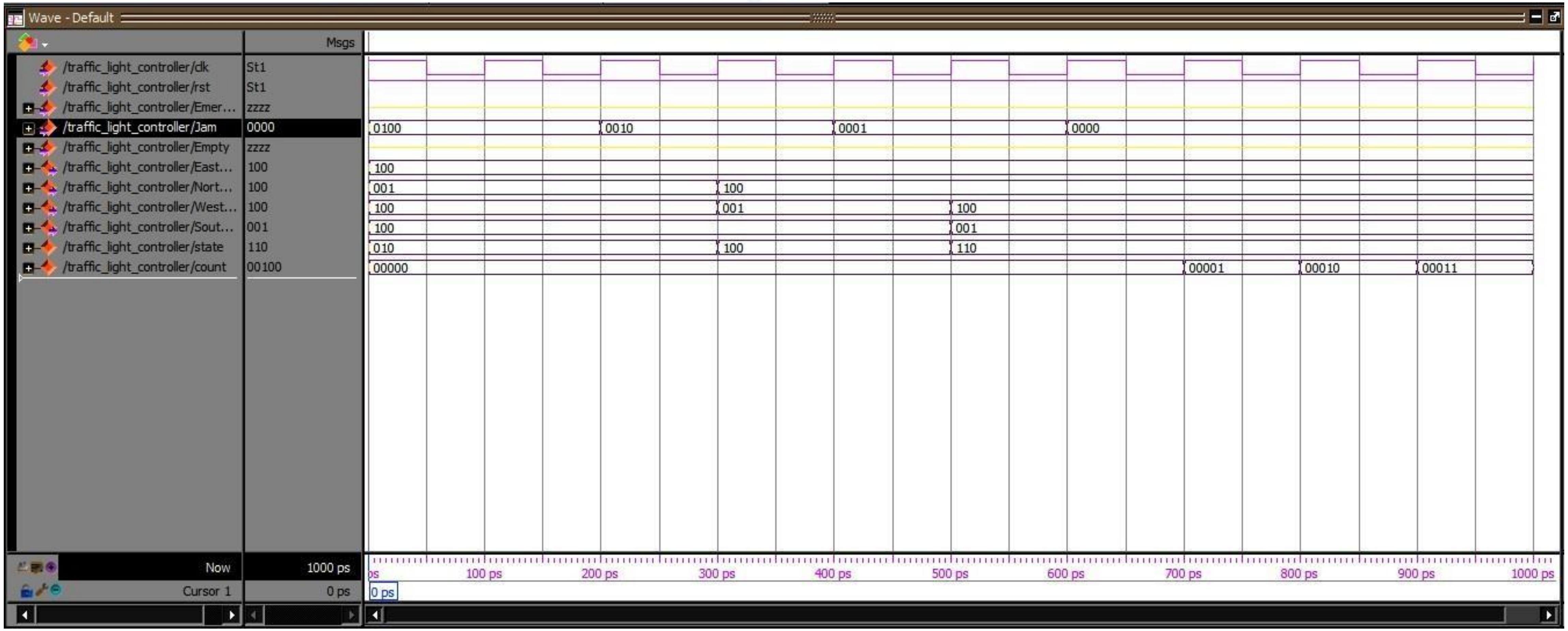
```
/traffic_light_controller/clk            St1
/traffic_light_controller/rst            St1
/traffic_light_controller/Emerge...      zzzz
/traffic_light_controller/Jam            zzzz
/traffic_light_controller/Empty          zzzz
/traffic_light_controller/East_road      010     001 ... 010
/traffic_light_controller/North_r...     110     100 ... 110
/traffic_light_controller/West_r...      100     100
/traffic_light_controller/South_r...     100     100
/traffic_light_controller/state          001     000 ... 001
/traffic_light_controller/count          00010   00101 00110 00111 01000 01001 01010 01011 01100 01101 01110 01111 10000 10001 10010 10011 00000 00001

Now       2200 ps
Cursor 1  0 ps

600 ps  800 ps  1000 ps  1200 ps  1400 ps  1600 ps  1800 ps  2000 ps  2200 ps  2400 ps
```

KONGU ENGINEERING COLLEGE

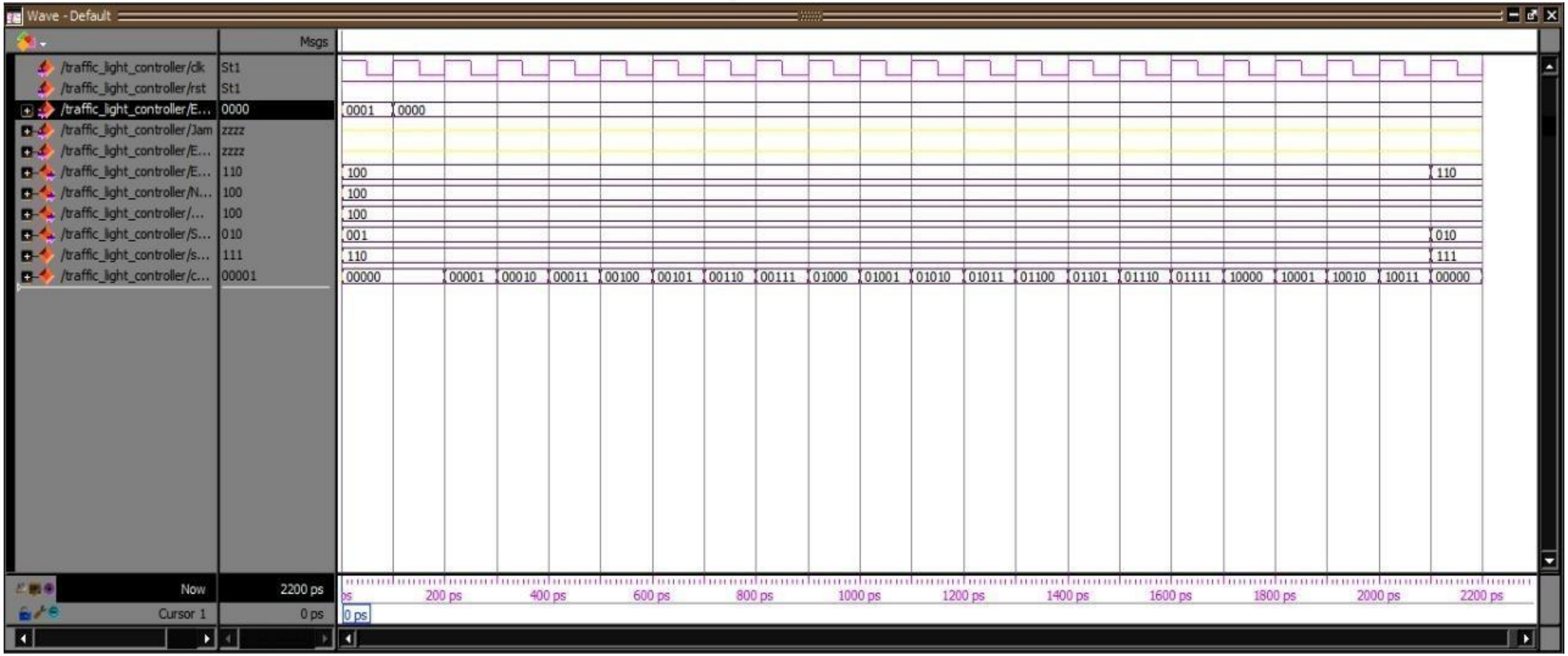
TRAFFIC JAM CONDITION



```
/traffic_light_controller/clk      St1
/traffic_light_controller/rst      St1
/traffic_light_controller/Emer...  zzzz
/traffic_light_controller/Jam      0000    0100 | 0010 | 0001 | 0000
/traffic_light_controller/Empty    zzzz
/traffic_light_controller/East...  100     100
/traffic_light_controller/Nort...  100     001 | 100
/traffic_light_controller/West...  100     100 | 001 | 100
/traffic_light_controller/Sout...  001     100 | 001
/traffic_light_controller/state    110     010 | 100 | 110
/traffic_light_controller/count    00100   00000 | 00001 | 00010 | 00011
```

KONGU ENGINEERING COLLEGE

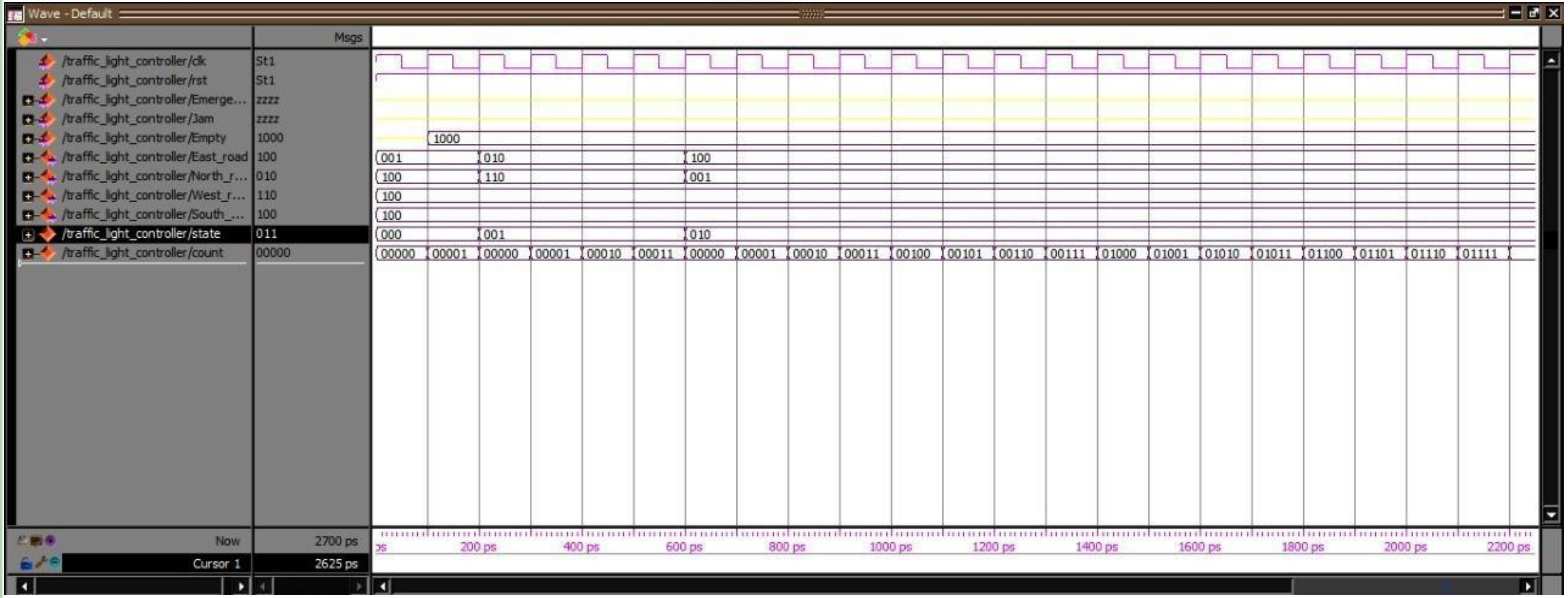
EMERGENCY CONDITION



```
Wave - Default
                                  Msgs
/traffic_light_controller/clk     St1
/traffic_light_controller/rst     St1
/traffic_light_controller/E...    0000    0001 | 0000
/traffic_light_controller/Jam     zzzz
/traffic_light_controller/E...    zzzz
/traffic_light_controller/E...    110     100 ... | 110
/traffic_light_controller/N...    100     100
/traffic_light_controller/...     100     100
/traffic_light_controller/S...    010     001 ... | 010
/traffic_light_controller/s...    111     110 ... | 111
/traffic_light_controller/c...    00001   00000 | 00001 | 00010 | 00011 | 00100 | 00101 | 00110 | 00111 | 01000 | 01001 | 01010 | 01011 | 01100 | 01101 | 01110 | 01111 | 10000 | 10001 | 10010 | 10011 | 00000

Now        2200 ps     ps  200 ps  400 ps  600 ps  800 ps  1000 ps  1200 ps  1400 ps  1600 ps  1800 ps  2000 ps  2200 ps
Cursor 1   0 ps        0 ps
```

EMPTY ROAD CONDITION



```
Wave - Default
                                   Msgs
/traffic_light_controller/clk      St1
/traffic_light_controller/rst      St1
/traffic_light_controller/Emerge...  zzzz
/traffic_light_controller/Jam      zzzz
/traffic_light_controller/Empty    1000
/traffic_light_controller/East_road  100
/traffic_light_controller/North_r...  010
/traffic_light_controller/West_r...  110
/traffic_light_controller/South_...  100
/traffic_light_controller/state    011
/traffic_light_controller/count    00000

Empty:      1000
East_road:  001  010  100
North_r:    100  110  001
West_r:     100
South:      100
state:      000  001  010
count:      00000 00001 00000 00001 00010 00011 00000 00001 00010 00011 00100 00101 00110 00111 01000 01001 01010 01011 01100 01101 01110 01111

Now       2700 ps
Cursor 1  2625 ps
ps  200 ps  400 ps  600 ps  800 ps  1000 ps  1200 ps  1400 ps  1600 ps  1800 ps  2000 ps  2200 ps
```

KONGU ENGINEERING COLLEGE

CONCLUSION

Thus an advanced traffic light control system for the convenience of the public during various conditions has been developed with Finite State Machine(FSM) using Verilog.

Thank You!