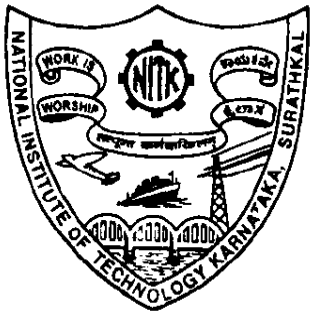
EC806 DSD using FPGA

LAB - 5



Report Submission

By

Inbasekaran Perumal

201EC226

Pranav Koundinya

201EC247

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, SURATHKAL

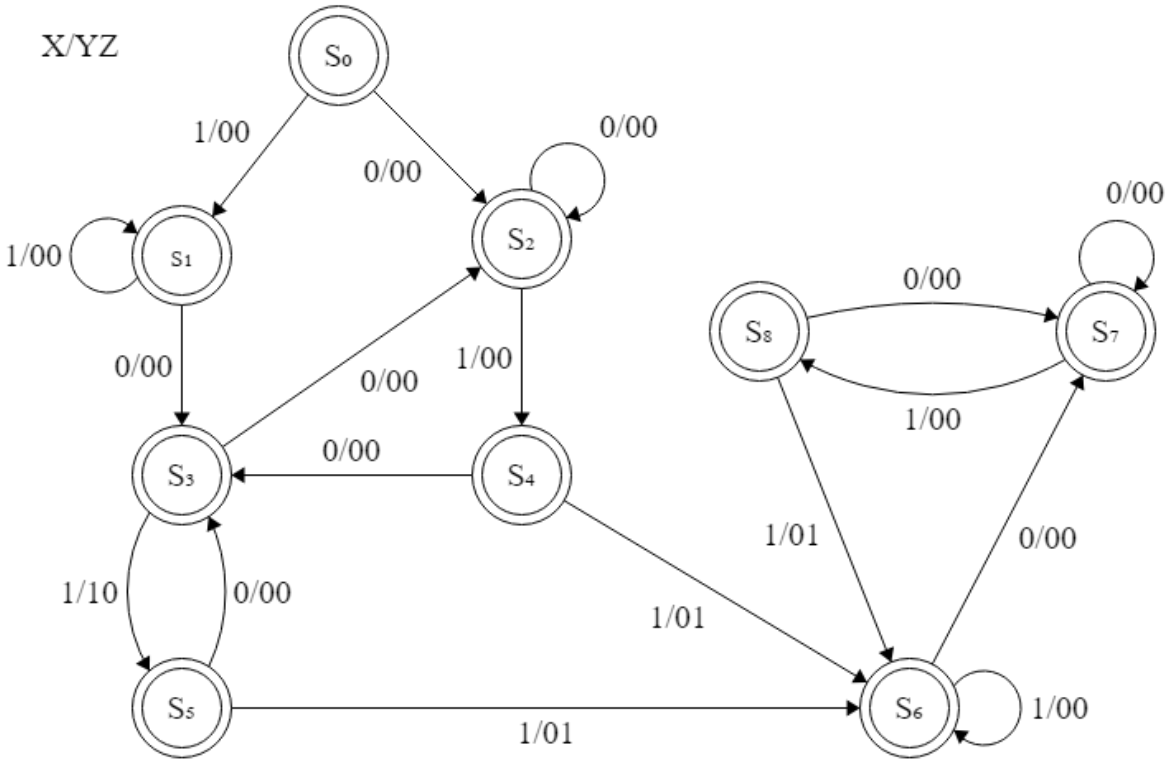
SRINIVASNAGAR 575025 KARNATAKA INDIA

11 October 2023

Q1) Sequence Detector

A digital system has one input X and two outputs Y and Z. The output Y=1 occurs every time the input sequence 101 is completed provided that the sequence 011 has never occurred. The output Z=1 occurs each time the output 011 is completed, i.e. once Z=1 has occurred, Y=1 can never occur but not vice versa. Design the Mealy state machine and implement it using Verilog.

State diagram:



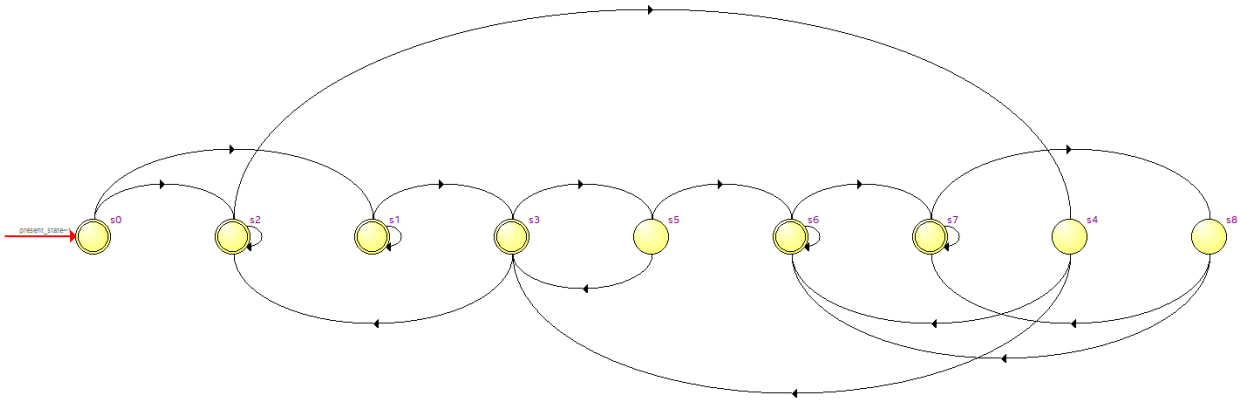
Verilog Code:

```
module sequence_detector(
    input x,
    input clk,
    input resetn,
```

```
    output reg Y,
    output reg Z
);

    parameter [3:0]s0 = 0, s1 = 1, s2 = 2, s3 = 3, s4 = 4, s5 = 5, s6 = 6, s7 = 7, s8
= 8;

    reg [3:0]present_state, next_state;
    reg y, z;

    // State Transition
    always @(negedge clk or negedge resetn)
    if(~resetn)
        present_state <= s0;
    else
        present_state <= next_state;

    // Registered output
    always @(negedge clk)
        begin
            Y <= y;
            Z <= z;
        end



    // Next State Decoder
    always @(present_state, x)
        case(present_state)
            s0 : if(x == 0) begin next_state = s2; {y, z} = 2'b00; end else begin
next_state = s1; {y, z} = 2'b00; end
            s1 : if(x == 0) begin next_state = s3; {y, z} = 2'b00; end else begin
next_state = s1; {y, z} = 2'b00; end
            s2 : if(x == 0) begin next_state = s2; {y, z} = 2'b00; end else begin
next_state = s4; {y, z} = 2'b00; end
            s3 : if(x == 0) begin next_state = s2; {y, z} = 2'b00; end else begin
next_state = s5; {y, z} = 2'b10; end
            s4 : if(x == 0) begin next_state = s3; {y, z} = 2'b00; end else begin
next_state = s6; {y, z} = 2'b01; end
            s5 : if(x == 0) begin next_state = s3; {y, z} = 2'b00; end else begin
next_state = s6; {y, z} = 2'b01; end
            s6 : if(x == 0) begin next_state = s7; {y, z} = 2'b00; end else begin
next_state = s6; {y, z} = 2'b00; end
            s7 : if(x == 0) begin next_state = s7; {y, z} = 2'b00; end else begin
next_state = s8; {y, z} = 2'b00; end
```

```
            s8 : if(x == 0) begin next_state = s7; {y, z} = 2'b00; end else begin
next_state = s6; {y, z} = 2'b01; end
            default : begin next_state = s0; {y, z} = 2'b00; end
        endcase

Endmodule
```

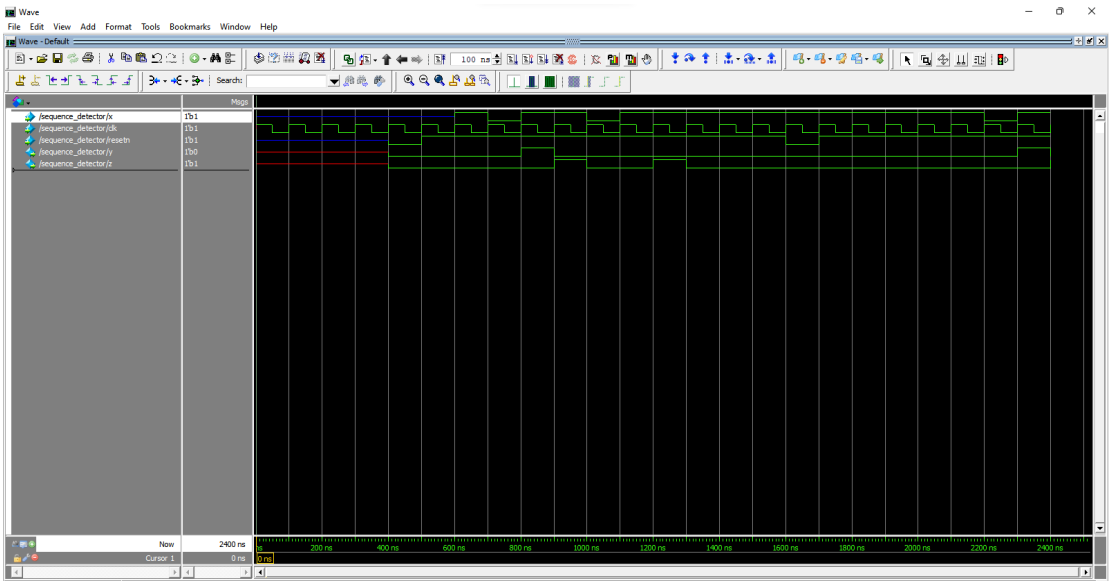
State Diagram generated by EDA:



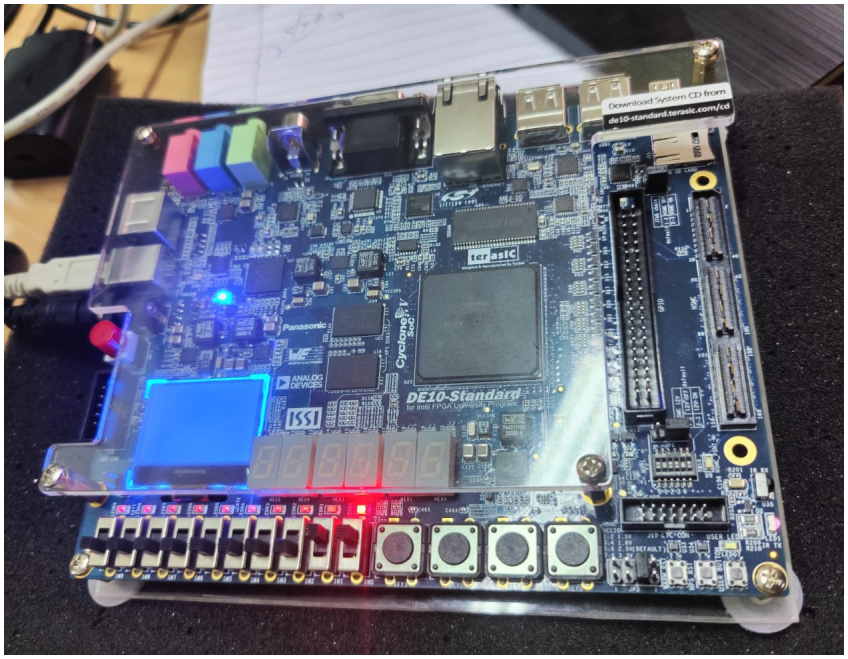
Device utilization:

Flow Status	In progress - Sat Oct 21 09:33:27 2023
Quartus Prime Version	22.1std.1 Build 917 02/14/2023 SC Lite Edition
Revision Name	seq_det
Top-level Entity Name	sequence_detector
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	5 / 41,910 (< 1 %)
Total registers	12
Total pins	5 / 499 (1 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

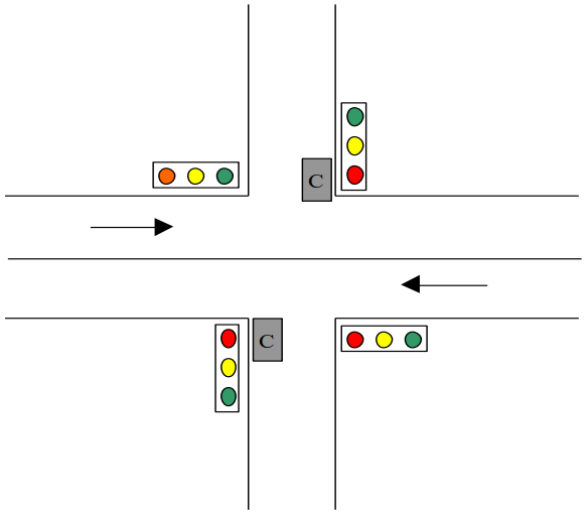
Simulation Results:



Results on FPGA:



Q2) Traffic light controller for NITK



Design Specification:

1. If no vehicles are detected on the NITK road, the traffic lights for the highway should stay green.
2. If a vehicle is detected on the NITK road, the highway lights should change from green to yellow and then to red. This allows the NITK road lights to turn green.
3. The NITK road lights should stay green only as long as a vehicle is detected on the NITK road and not longer than a set time interval. This is to ensure that traffic on the highway can flow.
4. Once these conditions are met, the NITK road lights should change from green to yellow and then to red, allowing the highway lights to turn back to green.
5. Even if vehicles are waiting to cross the highway, the highway lights should stay green for a set time interval.

Available Timers:

1. The timer is set using a control signal called ST (Set Timer).
2. Once the timer is set, it will send out a signal called TS after a short time interval has passed. This TS signal is used to time the yellow lights.
3. After a longer time interval, the timer will send out two more signals: TLH and TLN. These signals are used to time the green lights for the highway (TLH) and the NITK road (TLN), respectively.
4. The timer automatically resets itself whenever the ST signal is sent.

So, in simple terms, this timer helps control how long each light (green, yellow) stays on for both the highway and the NITK road based on these control signals (ST, TS, TLH, TLN). The timer resets every time it receives a new ST signal.
I hope this helps! Let me know if you have any other questions.

Inputs and outputs:

Input Signal	Description
reset	Place the FSM in the initial state
C	Car sensor on NITK road
TS	Short time interval expired for yellow light
TLH	Long time Interval expired for National highway
TLN	Long time interval expired for NITK road

Output Signal	Description
ST	Set timer for short and long intervals
highwayLights	Highway lights
nitkRoadLights	NITK lights

Output Light Signal Encoding	
GREEN	10
YELLOW	01
RED	00

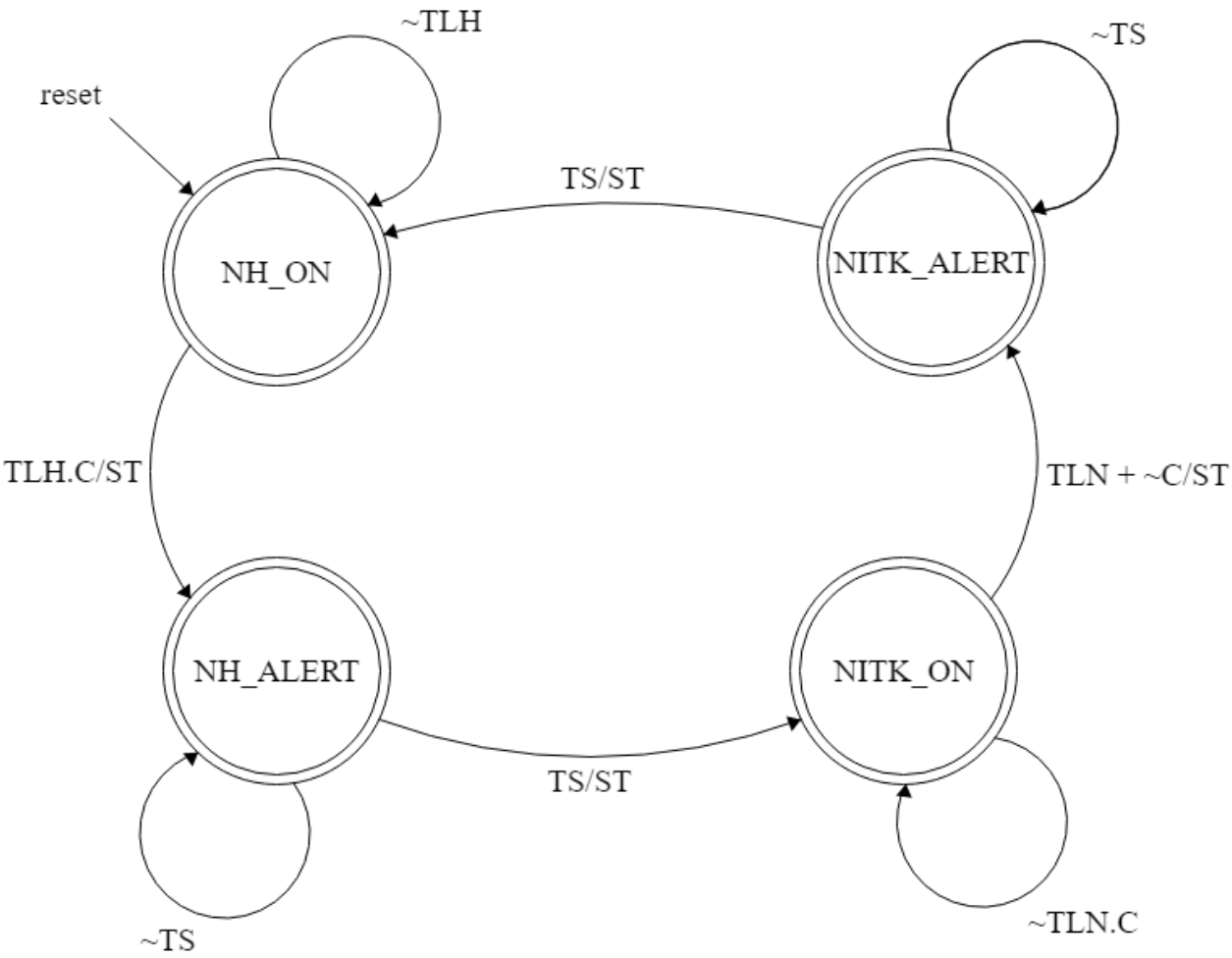
State	Description	
	National Highway	NITK Road

NH_ON	GREEN	RED
NH_ALERT	YELLOW	RED
NITK_ON	RED	GREEN
NITK_ALERT	RED	YELLOW

List of assumptions:

- Reset places the timer in NH_ON and starts the timer
- Stay in NH_ON as long there is no car on the NITK road
- Even if there is a car on the NITK road the highway stays on until the long time interval

State Diagram:



Verilog Code:

Top-level module

```
module TrafficLightControllerTop
#(
    parameter TS_VALUE = 2,
    TLH_VALUE = 5,
    TLN_VALUE = 4
)
(
    input wire clk, reset, C,
    output [1:0] highwayLights, // 00: Red, 01: Yellow, 10: Green
    output [1:0] nitkRoadLights // 00: Red, 01: Yellow, 10: Green
);
    wire TS, TLH, TLN,ST;
    wire reset_timer;
    assign reset_timer = reset|ST;
    Timer #(.TS_VALUE(TS_VALUE), .TLH_VALUE(TLH_VALUE), .TLN_VALUE(TLN_VALUE))
    timer(clk, reset_timer, TS, TLH, TLN);
    TrafficLightController trafficController
    (clk, reset, C, TS, TLH, TLN, ST, highwayLights, nitkRoadLights);

endmodule
```

Timer module

```
module Timer (
    input clk, reset,
    output reg TS, TLH, TLN
);

    reg [31:0] counter; // 32-bit counter

    // Assuming the clock frequency is 1Hz (one tick per second)
    parameter TS_VALUE = 2; // 2 seconds
    parameter TLH_VALUE = 5; // 5 seconds
    parameter TLN_VALUE = 4; // 4 seconds

    always @(posedge clk or posedge reset) begin
        if (reset) begin
```

```
            counter <= 32'b1;
            TS <= 1'b0;
            TLH <= 1'b0;
            TLN <= 1'b0;
        end else begin
            counter <= counter + 1;
            if (counter == TS_VALUE) begin
                TS <= 1'b1;
                TLN <= 1'b0;
                TLH <= 1'b0;
            end
            if (counter == TLN_VALUE) begin
                TLN <= 1'b1;
                TLH <= 1'b0;
                TS <= 1'b0;
            end
            if (counter == TLH_VALUE) begin
                TLH <= 1'b1;
                counter <= 32'b1; // Reset the counter after TLN_VALUE seconds
                TS <= 1'b0; // Reset the outputs
                TLN <= 1'b0;
            end
        end
    end

endmodule
```

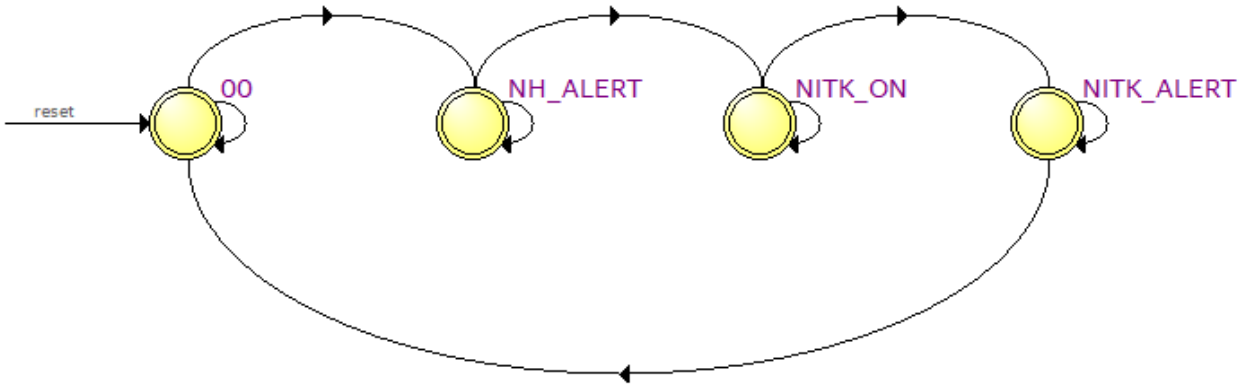
Traffic Light controller

```
module TrafficLightController(
    input wire clk, reset, C, TS, TLH, TLN,
    output reg ST,
    output reg [1:0] highwayLights, // 00: Red, 01: Yellow, 10: Green
    output reg [1:0] nitkRoadLights // 00: Red, 01: Yellow, 10: Green
);

    parameter GREEN = 2'b10;
    parameter YELLOW = 2'b01;
    parameter RED = 2'b00;

    localparam NH_ON=2'b00, NH_ALERT=2'b01, NITK_ON=2'b10, NITK_ALERT=2'b11;
    reg [1:0] present_state=0, next_state=0;
```

```
// Present state register
always @(posedge clk or posedge reset) begin: L_PRESENT_STATE_REGISTER
    if(reset) begin
        present_state <= NH_ON;
    end else begin
        present_state <= next_state;
    end
end

// Next state logic
always@(present_state or C or TLH or TLN or TS) begin:L_NEXT_STATE_LOGIC
    casex(present_state)
        NH_ON: next_state = C&&TLH?NH_ALERT:NH_ON;
        NH_ALERT: next_state = TS?NITK_ON:NH_ALERT;
        NITK_ON: next_state = !C||TLN?NITK_ALERT:NITK_ON;
        NITK_ALERT: next_state = TS?NH_ON:NITK_ALERT;
        default: next_state = NH_ON;
    endcase
end

// Output decoder
always@(present_state) begin:L_OUTPUT_DECODER
    casex(present_state)
        NH_ON: begin
            highwayLights = GREEN;
            nitkRoadLights = RED;
            ST = C&&TLH;
        end
        NH_ALERT: begin
            highwayLights = YELLOW;
            nitkRoadLights = RED;
            ST = TS;
        end
        NITK_ON: begin
            highwayLights = RED;
            nitkRoadLights = GREEN;
            ST = TLN||!C;
        end
        NITK_ALERT: begin
            highwayLights = RED;
            nitkRoadLights = YELLOW;
            ST = TS;
        end
        default: begin
            highwayLights = 2'b11;
```

```
                nitkRoadLights = 2'b11;
                ST = 0;
            end
        endcase
    end

endmodule
```

State Diagram generated by EDA



	Source State	Destination State	Condition
1	00	NH_ALERT	(C).(TLH)
2	00	00	(!C) + (C).(!TLH)
3	NH_ALERT	NH_ALERT	(!TS)
4	NH_ALERT	NITK_ON	(TS)
5	NITK_ALERT	NITK_ALERT	(!TS)
6	NITK_ALERT	00	(TS)
7	NITK_ON	NITK_ALERT	(!C) + (C).(TLN)
8	NITK_ON	NITK_ON	(C).(!TLN)

Device Utilization:

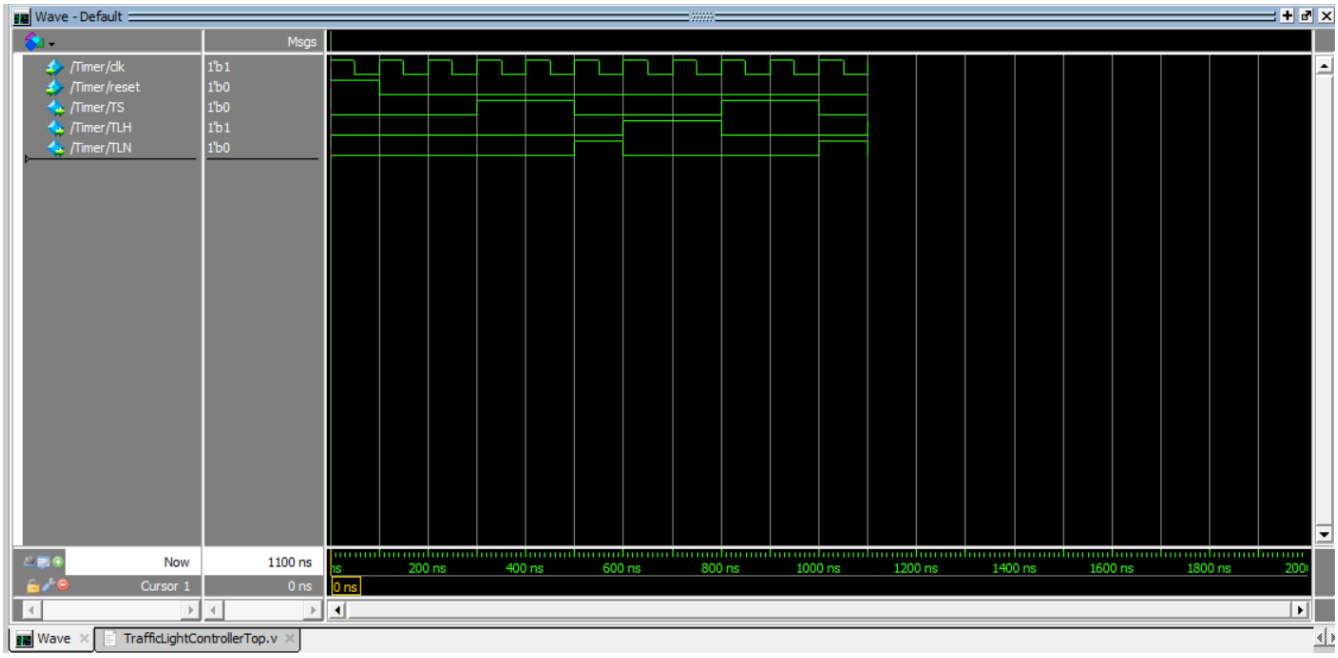
Flow Summary

<<Filter>>

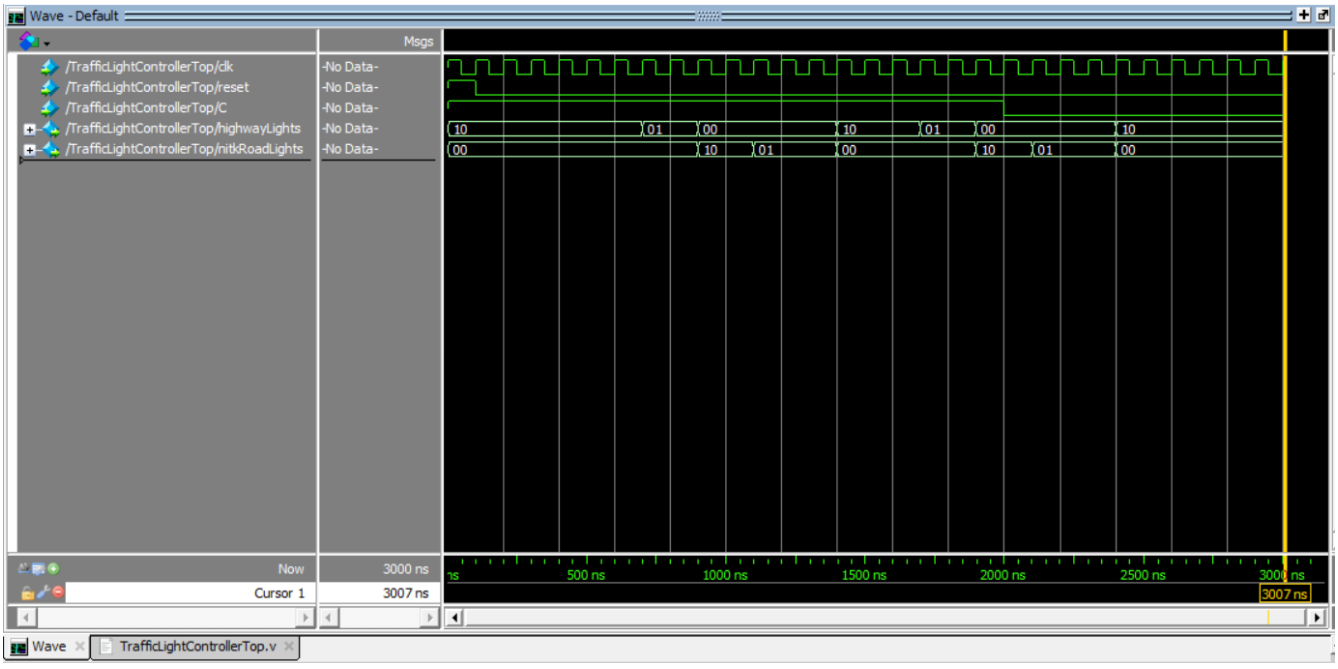
Flow Status	Successful - Sat Oct 21 09:29:42 2023
Quartus Prime Version	22.1std.1 Build 917 02/14/2023 SC Lite Edition
Revision Name	lab5
Top-level Entity Name	TrafficLightControllerTop
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	32 / 41,910 (< 1 %)
Total registers	60
Total pins	7 / 499 (1 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

Simulation Results:

Traffic control timer



Traffic Control Top-level module



Results on FPGA: