VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

LSI LOGIC DESIGN

LAB 1

Simulation

Group 6
Võ Trung Kiên - 2153502

HO CHI MINH CITY, March 2024

Contents

1 Customer requirement

1.1 Specification

In this exercise, you must create RTL code for the bound flasher with 16 lamps which has operation as below:

At the initial state, all lamps are OFF. If flick signal is ACTIVE (set 1), the flasher start operating:

- The lamps are turned ON gradually from lamp[0] to lamp[5].
- The lamps are turned OFF gradually from lamp[5] (max) to lamp[0] (min).
- The lamps are turned ON gradually from lamp[0] to lamp[10].
- The lamps are turned OFF gradually from lamp[10] (max) to lamp[5] (min).
- The lamps are turned ON gradually from lamp[5] to lamp[15].
- The lamps are turned OFF gradually from lamp[15] to lamp[0].
- Finally, the lamps are turned ON then OFF simultaneously (blink), return to the initial state.

Additional condition:

- At each kickback point (lamp[5] and lamp[10]), if flick signal is ACTIVE, the lamps will turn OFF gradually again to the min lamp of the previous state, then continue operation as above description.
- For simplicity, kickback points are considered only when the lamps are turned ON gradually, except the first state.

1.2 Example of specification

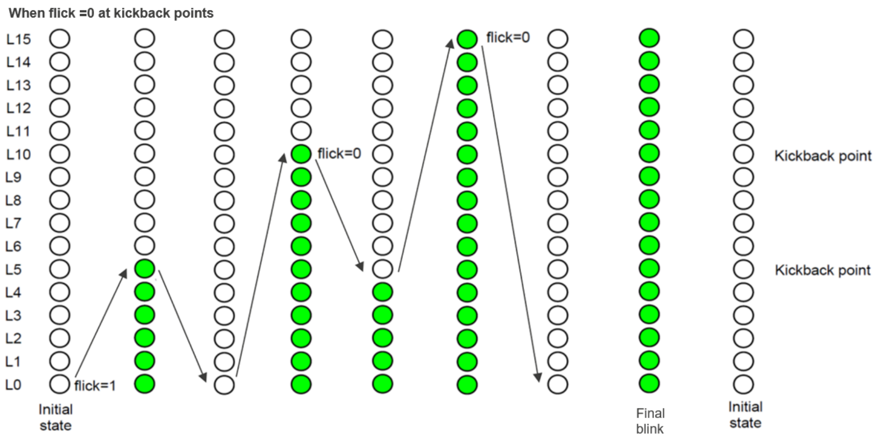

Figure 1. When flick = 0 at kick back point

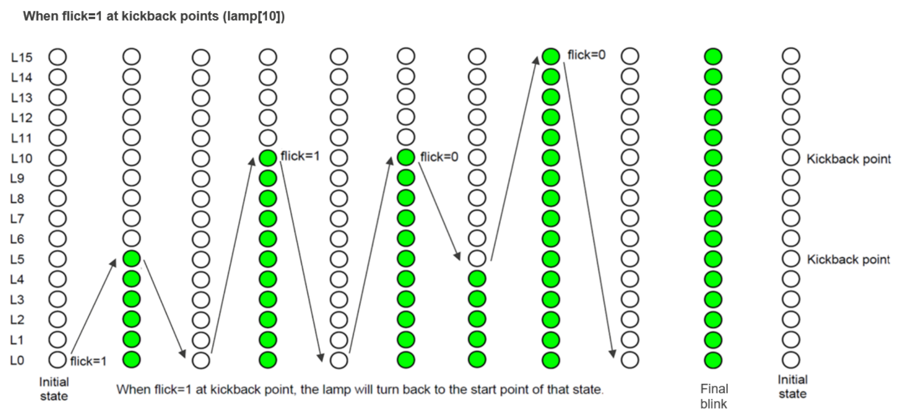
When flick=1 at kickback points (lamp[10])
L15
L14
L13
L12
L11
L10
L9
L8
L7
L6
L5
L4
L3
L2
L1
L0
flick=1
flick=1
flick=0
flick=0
Kickback point
Kickback point
Initial state
When flick=1 at kickback point, the lamp will turn back to the start point of that state.
Final blink
Initial state

Figure 2. When flick = 1 at kick back point lamp[10]

2 Logic Design

2.1 Signal/Width/In or Out

```
input wire flick;
input wire clk;
input wire rst;
output reg [15:0] led_output;

reg [3:0] state;
reg [3:0] stateR;
reg [15:0] led_buffer;
```

2.2 Finite State Machine Design

2.2.1 Control Lamps At State

```
always @(*) begin
        case(state)
            3'b000: begin
                led_buffer <= 16'b0;
            end

            3'b001: begin
                if (led_output[5] != 1) begin
                    led_buffer <= (led_output << 1) | 1'b1;
                end
            end

            3'b010: begin
                if (led_output[0] != 0) begin
                    led_buffer <= (led_output >> 1);
                end
            end
            3'b011: begin
                if (led_output[10] != 1) begin
                    led_buffer <= (led_output << 1) | 1'b1;
                end
            end

            3'b100: begin
                if (led_output[5] != 0) begin
                   led_buffer <= (led_output >> 1);
                end
            end

            3'b101: begin
                 if (led_buffer[15] != 1) begin
                     led_buffer <= (led_output << 1) | 1'b1;
                 end
            end

            3'b110: begin
                 if (led_output[0] != 0) begin
                    led_buffer <= (led_output >> 1);
                 end
            end

            3'b111: begin
                    led_buffer <= 16'b1111111111111111;
            end

            4'b1000: begin // State KickBack lamp[5] or lamp[10] at state 3
                if (led_output[0] != 0) begin
                    led_buffer <= (led_output >> 1);
                end
            end

            4'b1001: begin // State KickBack lamp[5] or lamp[10] at state 5
                if (led_output[5] != 0) begin
                    led_buffer <= (led_output >> 1);
                end
            end

            default: begin
                        led_buffer <= 16'b0;
                     end
        endcase
    end
```

2.2.2 Control State

```
always @(*) begin
         if (rst == 1'b0) begin
           state <= 3'b000;
         end
         else begin
         case (stateR)
           3'b000: begin //INITIAL
             if (flick == 1) begin
               state <= 3'b001;
             end
           end

           3'b001: begin //STATE_1
             if (led_output[5] == 1) begin
               state <= 3'b010;
             end
           end

           3'b010: begin    //STATE_2
             if (led_output[0] == 0) begin
                  state <= 3'b011;
             end
           end

           3'b011: begin //STATE_3
             if ((led_output[10] == 1 && flick == 0)) begin
                  state <= 3'b100;
             end
             else if((flick == 1 && led_output[5] == 1 && led_output[6] == 0) || (flick == 1 && l
                  state <= 4'b1000; //kickback at lamp[5] or lamp[10]
             end
           end

           3'b100: begin    //STATE_4
             if (led_output[5] == 0) begin
                  state <= 3'b101;
             end
           end

           3'b101: begin //STATE_5
             if ((led_output[15] == 1 && flick == 0) || (led_output[15] == 1 && flick == 1))
                  state <= 3'b110;
             else if(((flick == 1) && (led_output[5] == 1) && (led_output[6] == 0)) || ((flick ==
                  state <= 4'b1001;  //kickback at lamp[5] or lamp[10]
             end
           end
```

```
    4'b1000: begin // kick back at lamp [5] or lamp[10] at state 3
     if(led_output[0] == 0) begin
         state <= 3'b011;
     end
    end
    4'b1001: begin // kick back at lamp [5] or lamp[10] at state 5
     if(led_output[5] == 0) begin
         state <= 3'b101;
     end
    end

    3'b110: begin //SEMI-FINAL
      if (led_output[0] == 0) begin
           state <= 3'b111;
      end
    end

    3'b111: begin //FINAL
         if (led_output[0] == 1 && led_output[15] == 1)
             state <= 3'b000;
    end

    default: state <= 3'b000;
  endcase
  end
end
```

3 Logic Verification

3.1 Normal Flow

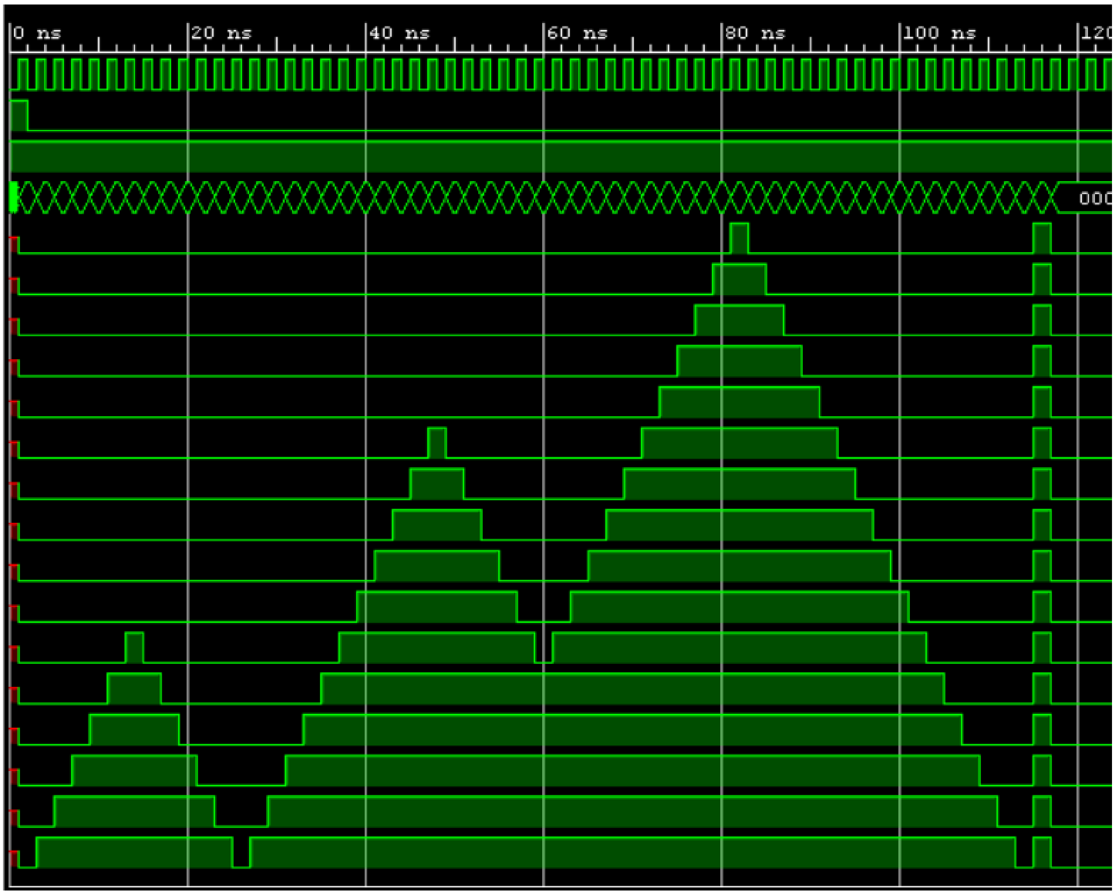

Figure 3. Normal Flow

Description: Display leds according to the normal pattern without reset flick signal (flick = 0 in the entire run).

3.2 Flick signal at L5 in state 3

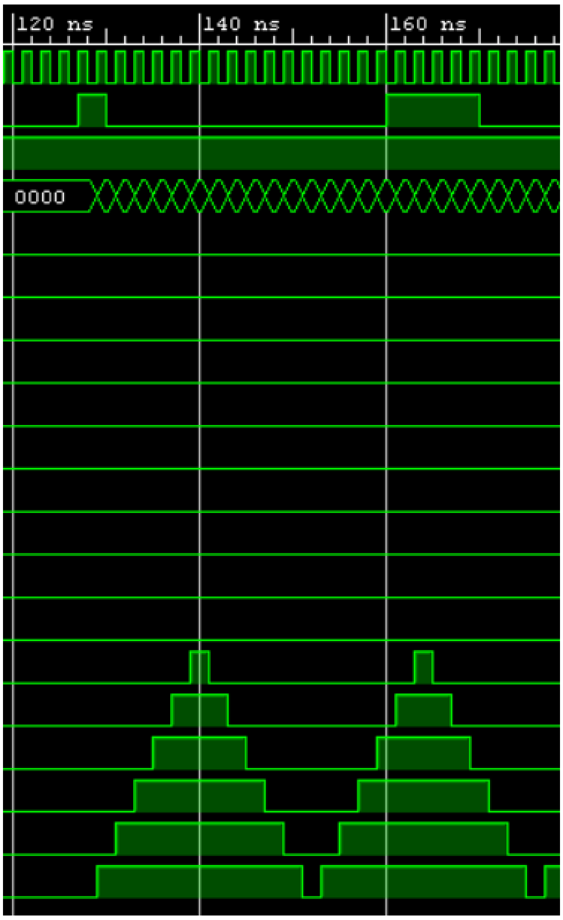

Figure 4. Flick signal at L5 in state 3

Description: When led L5 turn on at state 2 and if flick signal triggers, system goes back to the previous state, turning off leds from L5 to L0, then turning on leds from L0 to L10.

3.3 Flick signal at L10 in state 3

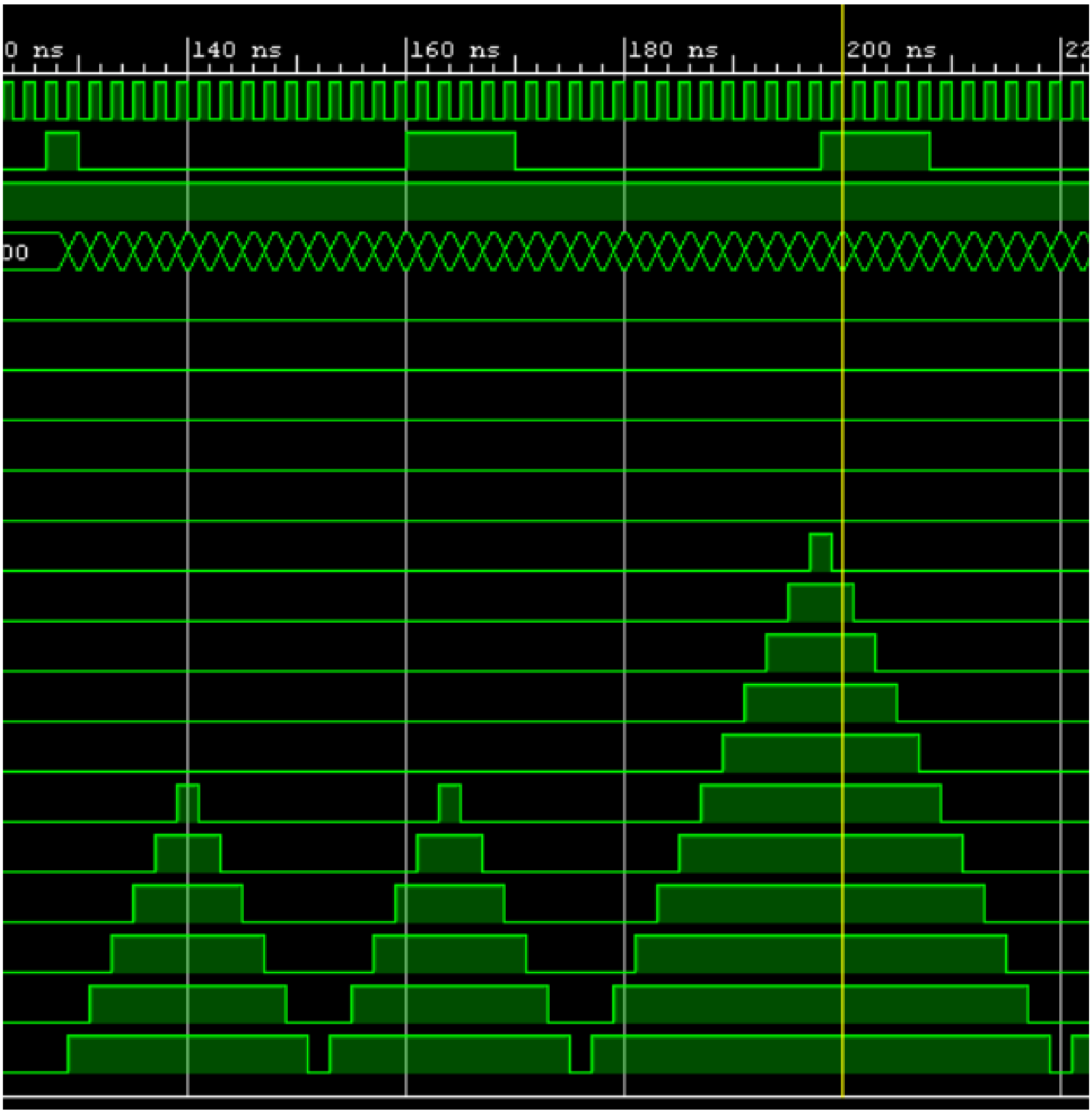

Figure 5. Flick signal at L10 in state 3

Description: When led L10 turn on at state 3 and if flick signal triggers, system goes back to the previous state, turning off leds from L10 to L0, then turning on leds from L0 to L10.

3.4 Flick signal at L5 and L10 in state 5

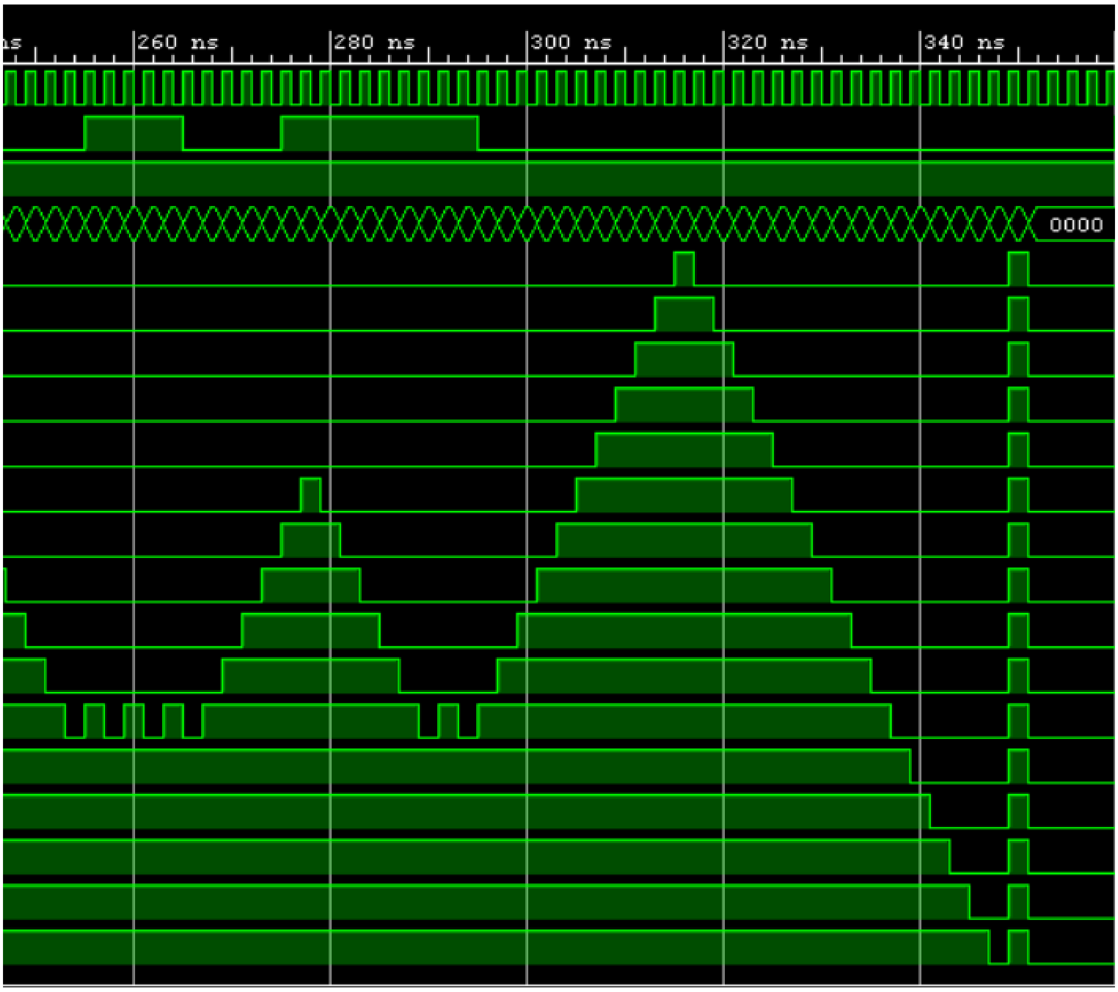

Figure 6. Flick signal at L5 and L10 in state 5

Description:

- When led L5 turn on at state 5 and if flick signal triggers, system goes back to the previous state, turning off leds L5, then turning on leds from L5 to L15.
- When led L10 turn on at state 5 and if flick signal triggers, system goes back to the previous state, turning off leds from L10 to L5, then turning on leds from L5 to L15.

3.5 Flick signal to repeat the process

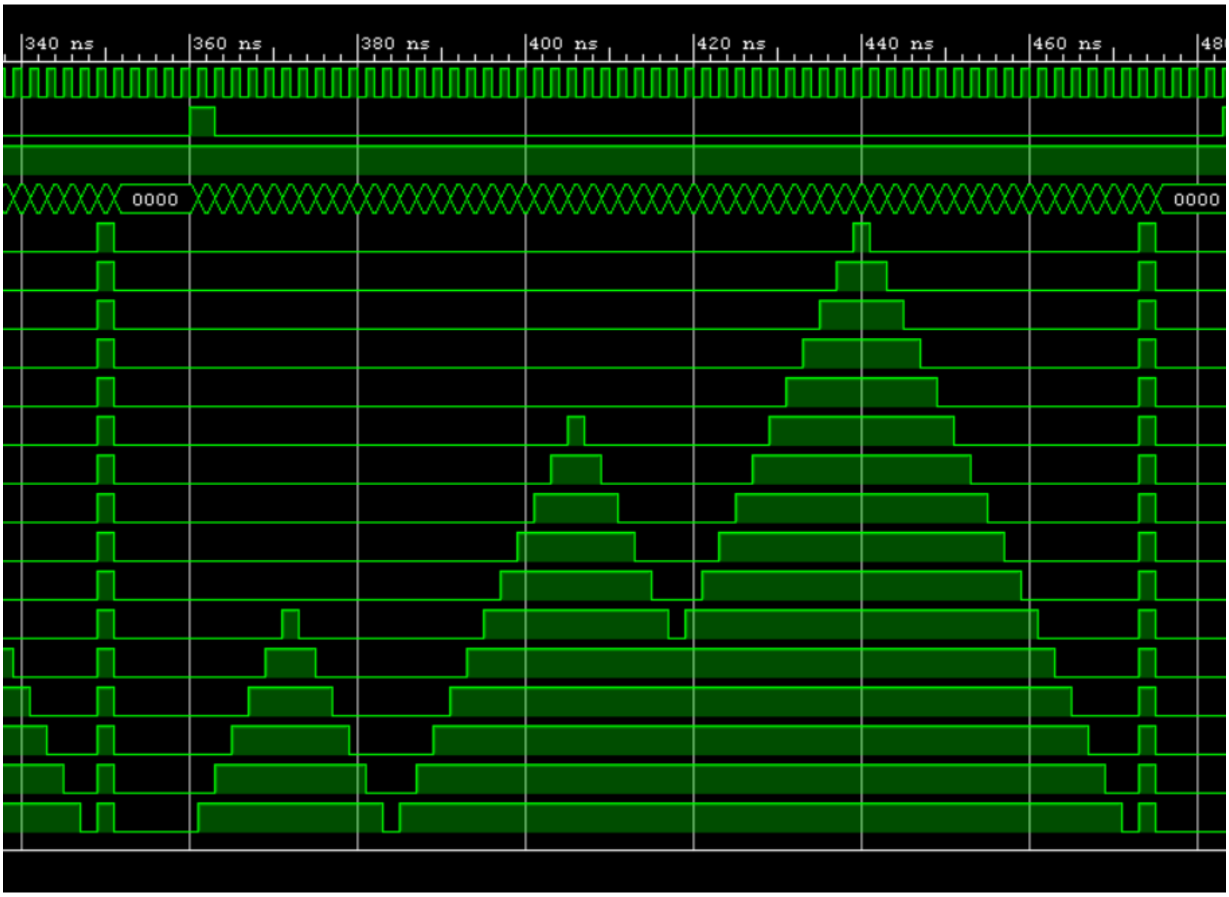

Figure 7. Flick signal to repeat the process

Description: After a complete flow, flick signal to check the process can be repeated from state 1 not state 6.

3.6 Flick signal at any time slot (not kickback point in turning off LED state) && Flick signal at between L5 and L10 in state 3 && Reset signal at any time slot

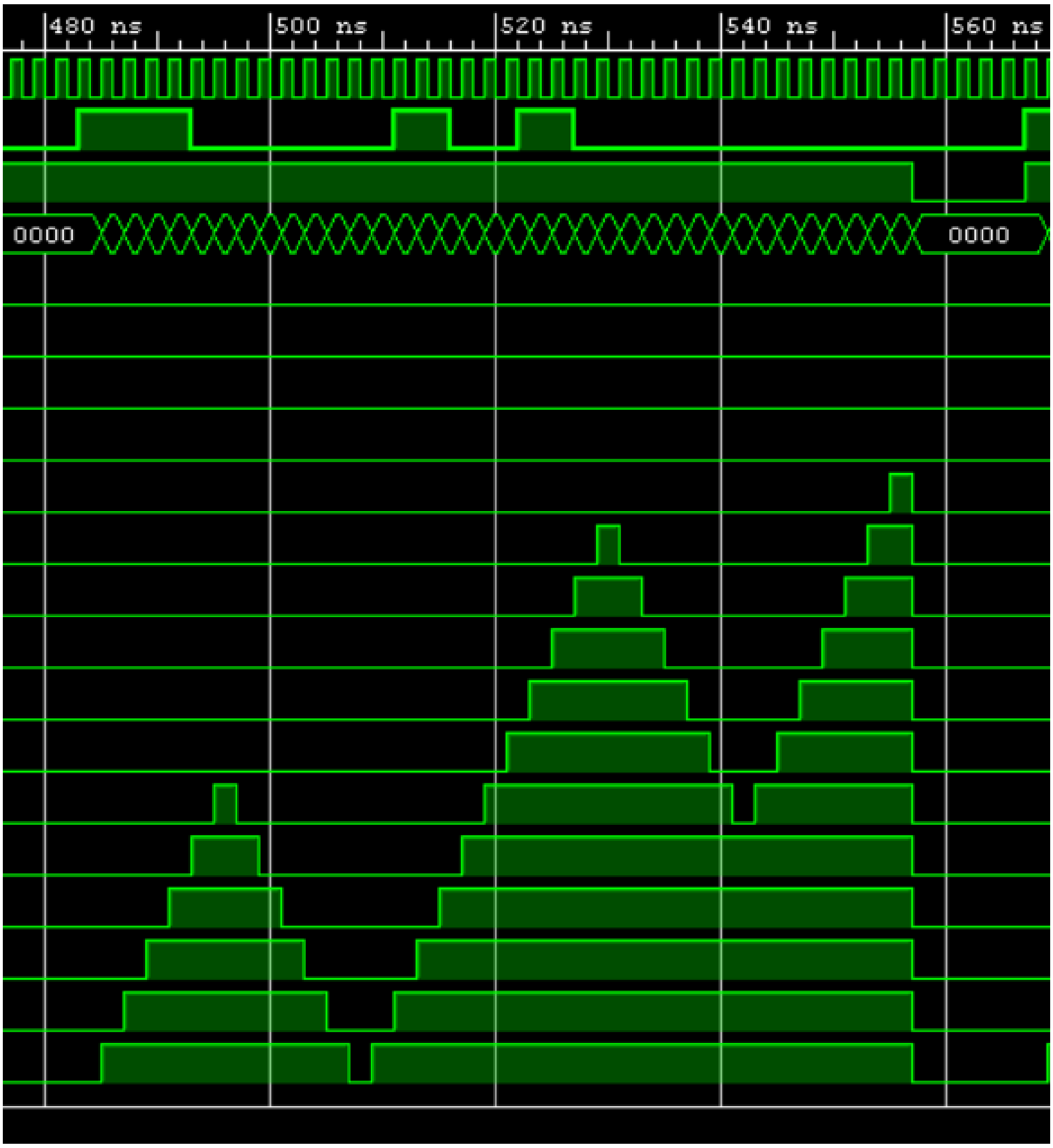

Figure 8. Flick signal at any time slot (not kickback point in turning off LED state) && Flick signal at between L5 and L10 in state 3 && Reset signal at any time slot

Description:

- If flick signal takes effect in those states then the code is wrong, else then everything is

OK.

- Between the kickback point L5 and L10, trigger flick signal to test if there is any state transition, if yes, then the code is wrong, else then the code working normally.
- Whenever reset signal is 0, the system reset immediately to the Initial State, all leds are off.

3.7 Reset signal with flick signal at kickback point in turning off leds state

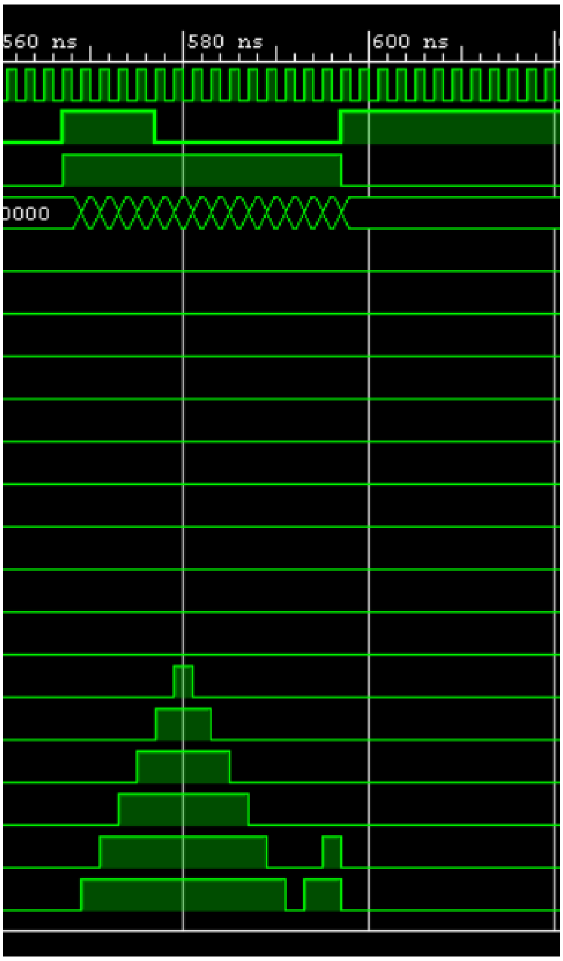

Figure 9. Reset signal with flick signal at kickback point in turning off leds state

Description: Both reset signal and flick signal trigger at the same time, system must go back to Initial state.

4 Simulation

After design and verification I use Cadence tool to check the simulation as I show below.

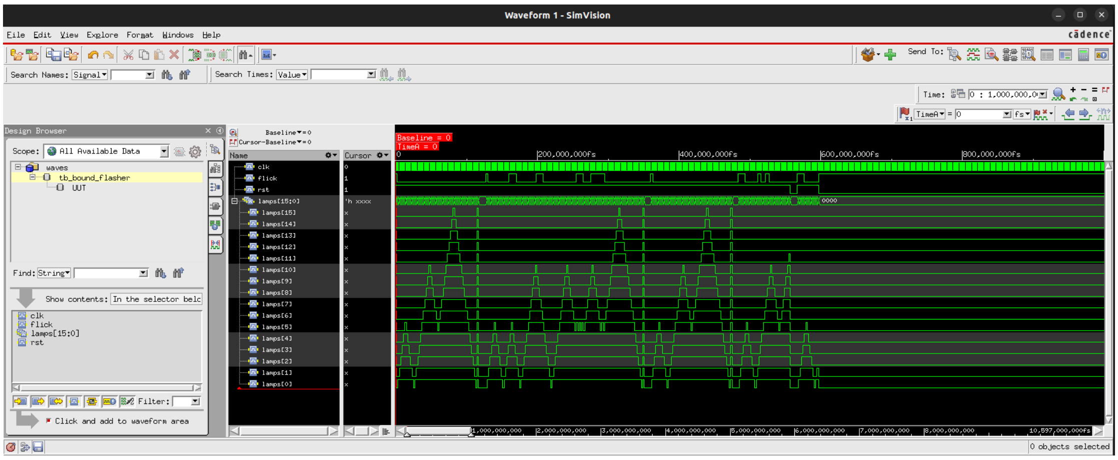

Figure 10. Simulation all test cases after design and verification

In conclusion, in this lab it can familiarize me with one of Cadence simulation tool - Xcelium.

This Lab can cover me the following:

- How to use Xcelium for execute simulation.
- How to generate Waveform file by Xcelium.
- How to debug by using GUI (Graphic User Interface).