VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

SEMESTER 232 - CO3098 - CC02

LSI LOGIC DESIGN

REPORT LABORATORY 1 - BOUND FLASHER

Under the guidance of: Prof. Thien-An Nguyen

Student: Nguyen Van Thanh Dat - 2152055

HO CHI MINH CITY, APRIL 2024

Contents

1 Specification

Create RTL code for the Bound Flasher with 16 lamps which has operation as below:

- At the initial state, all lamps are OFF. If the flick signal is ACTIVE (set 1), the flasher starts operating:
  1. The lamps are turned ON gradually from lamp[0] to lamp[5].
  2. The lamps are turned OFF gradually from lamp[5] (max) to lamp[0] (min).
  3. **The lamps are turned ON gradually from lamp[0] to lamp[10].**
  4. The lamps are turned OFF gradually from lamp[10] (max) to lamp[5] (min).
  5. **The lamps are turned ON gradually from lamp[5] to lamp[15].**
  6. The lamps are turned OFF gradually from lamp[15] to lamp[0].
  7. Finally, the lamps are turned ON then OFF simultaneously (blink), and return to the initial state.
- **Additional condition:** At each kickback point (*lamp[5]* and *lamp[10]*), if the 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 in the first state.

2 Implementation

Based on the given specification, we can utilize some concepts with RTL code to implement the Bound Flasher:

```
`timescale 1ns / 1ps
//////////////////////////////////////////
// Company: HCMUT
// Engineer: Nguyen Van Thanh Dat
//
// Create Date: 03/03/2024 02:28:06 PM
// Design Name: BoundFlasherLab1
// Module Name: BoundFlasher
// Project Name: LAB1
// Revision: Prof. Nguyen Thien An
// Revision 0.01 - File Created
//////////////////////////////////////////////

module boundFlasher(
    input clk, flick, rst,
    output reg [15:0] leds
);

    parameter s0 = 0;
    parameter s1 = 1;
    parameter s2 = 2;
    parameter s3 = 3;
```

```
    parameter s4 = 4;
    parameter s5 = 5;
    parameter s6 = 6;
    parameter s7 = 7;

    parameter timing = 100;

    integer counter = 0;

    reg [2:0] state = s0; //**NOTE: LAB 3. JUST NEED TO CHANGE reg from [3:0] to [2:0], then everything is good

    always @(posedge clk, posedge rst) begin
        if (rst) begin
         state <= s0;
         counter <= 0;
        end

        else begin
        case(state)
            s0:
                begin
                    leds <= 16'h0000;
                    if(flick == 0) state <= s0;
                    else state <= s1;
                end
            s1:
                begin
                    counter <= counter + 1;
                    if(counter == timing) begin
                        leds <= (leds << 1) + 1; //gradually ON
                        counter <= 0;
                    end
                    if (leds == 16'h003F) begin //L[5]
                        state <= s2;
                        counter <= 0;
                    end
                end
            s2:
                begin
                    counter <= counter + 1;
                    if (counter == timing) begin
                        leds <= leds >> 1; //gradually OFF
                        counter <= 0;
                    end
                    if (leds == 16'h0000) begin //-> L[0]
                        state <= s3;
                        counter <= 0;
```

```
70                      end
71                  end
72              s3:
73                  begin
74                      counter <= counter + 1;
75                      if (counter == timing) begin
76                          leds <= (leds << 1) + 1; //L[0] -> L[5]* -> L
    [10]*
77                          counter <= 0;
78                      end
79
80                      if (leds == 16'h003F) begin
81
82                          if(flick == 1) begin
83                          state <= s2;
84                          counter <= 0;
85                          end
86                      end
87
88                      else if (leds == 16'h07FF) begin
89                           counter <= 0;
90                           if(flick == 1) state <= s2;
91                           else state <= s4;
92                       end
93                  end
94              s4:
95                  begin
96                      counter <= counter + 1;
97                      if(counter==timing) begin
98                          leds <= leds >> 1;
99                          counter <= 0;
100                     end
101                     if (leds == 16'h001F) begin // L[5]
102                         counter<=0;
103                         state <= s5;
104                     end
105                 end
106
107             s5:
108                 begin
109                     counter <= counter + 1;
110                     if (counter == timing) begin
111                         counter <= 0;
112                         leds <= (leds << 1) +1; //case, increase to L[5]
    and check KBP
113                     end
114                     if (leds == 16'h003F || leds == 16'h07FF) begin
115                         if (flick == 1) begin
```

```
116                         counter <=0;
117                         state <= s4;
118                         end
119                     end
120                     if (leds == 16'hFFFF && flick == 0) begin
121                         counter <=0;
122                         state <= s6;
123                     end
124                 end
125 
126             s6:
127                 begin
128                     counter <= counter + 1;
129                     if (counter == timing) begin
130                         counter <= 0;
131                         leds <= leds >> 1; //OFF to L[0]
132                     end
133                     if(leds == 16'h0000) begin
134                         counter <= 0;
135                         state <= s7;
136                     end
137                 end
138             s7:
139                 begin //ON and OFF SIMULTANEOUSLY
140                     leds <= 16'hFFFF;
141                     counter <= counter + 1;
142                     if (counter == timing) begin
143                         counter <= 0;
144                         state <= s0;
145                     end
146 
147                 end
148         endcase
149     end
150   end
151 endmodule
```

3 Simulation

Here is the test bench file for the Bound Flasher for further simulation:

```
module boundFlasher_tb;
    reg clk;
    reg flick;
    reg rst;
    wire [15:0] leds;

    boundFlasher UUT
    (
        clk, flick, rst, leds
    );

    // Create clock
    always #1 clk = !clk;

    initial begin
      // Initial state
        clk = 0;
        flick = 0;
        rst = 0;
        #10;
        flick = 1;
        @(UUT.state == 2) rst =1;
        #20
      // flick = 1 -> system starts operating
      rst=0;
        flick = 1;
        #10;
        flick = 0;

      // Test kickback points at L[5] at state 3
      @(UUT.state == 3)
         begin
            #10;
            flick = 1;
         end
      // Set flick = 1 to transit to next state
      @(UUT.state == 2) flick = 0;
      // Test kickback points at L[10] at state 3
      @(UUT.leds == 16'h007F) flick=1;
      @(UUT.state == 2) flick = 0;


      // Test kickback points at L[5] at state 5
      @(UUT.state == 5) flick = 1;
```

```
45        @(UUT.state == 4) flick = 0;
46        // Test kickback points at L[10] at state 5
47        @(UUT.leds == 16'h007F) flick=1;
48          @(UUT.state == 4) flick = 0;
49        #50000;
50        $finish;
51      end
52
53      initial begin
54        $recordfile ("waves");
55        $recordvars ("depth=0", boundFlasher_tb);
56      end
57
58  endmodule
```

In the beginning, it is immediately evident that the system will not operate in case there is no flick signal. Then, we activate the flick signal in order to make the Bound Flasher start operating:

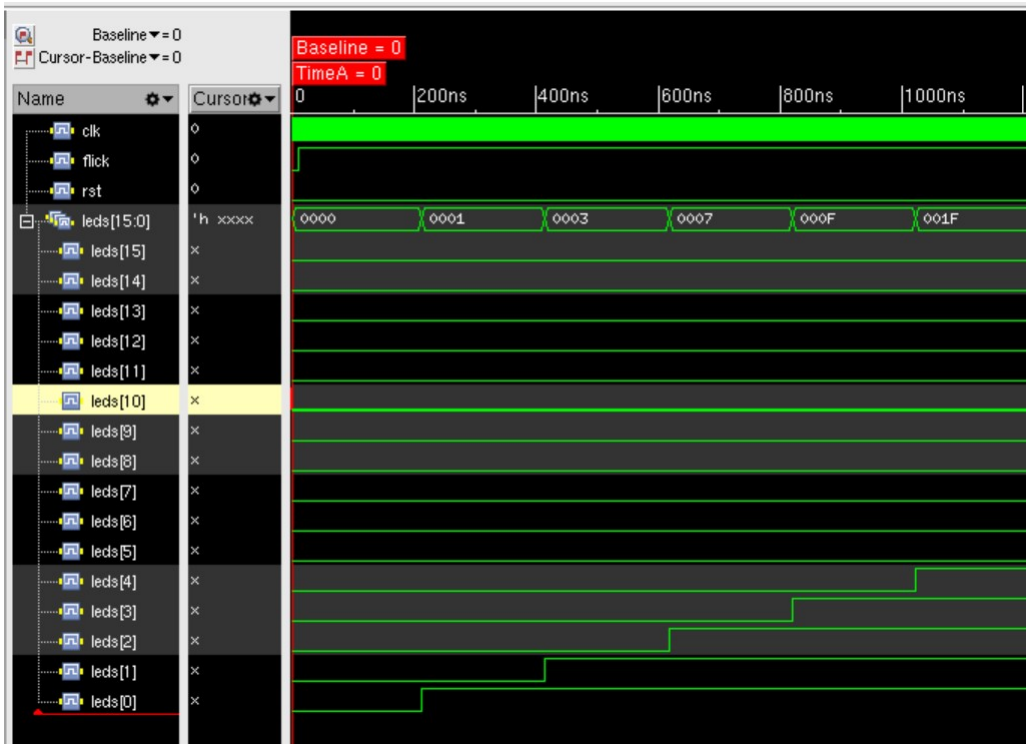

Figure 1: Flick signal case

Moving on to the reset (rst), this will make the system reset straight away whenever users activate the reset function:

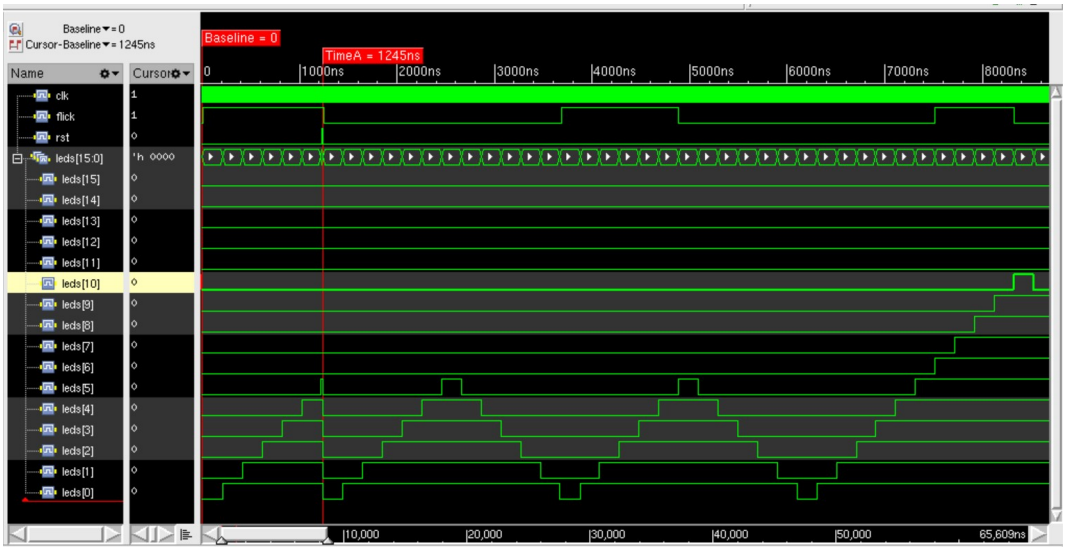

Figure 2: Reset case

Upon to the given specification mentioned initially, we will simulate to check the state of the LEDs in kickback points with active flick signal (LEDs[5] and LEDs[10] in the situations where the lamps are turned ON gradually from lamp[0] to lamp[10] and the lamps are turned ON gradually from lamp[5] to lamp[15]) and also the system runs without active flick signal when encountering kickback points as mentioned:

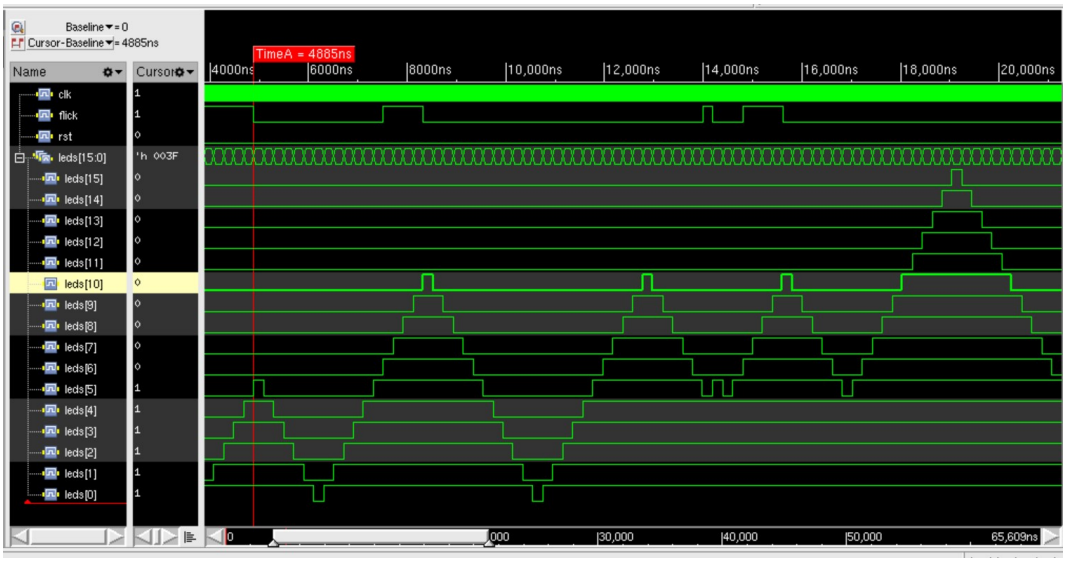

Figure 3: Final simulation