RTL_EXERCISE_1 BOUND FLASHER

Author	- Nguyễn Tuấn Anh - Ung Ngô Minh Lăng - Tăng Văn Minh - Lê Minh Phúc - Đặng Đình Thông
Date	2023/03/2
Version	1.1

Contents

1. Interface

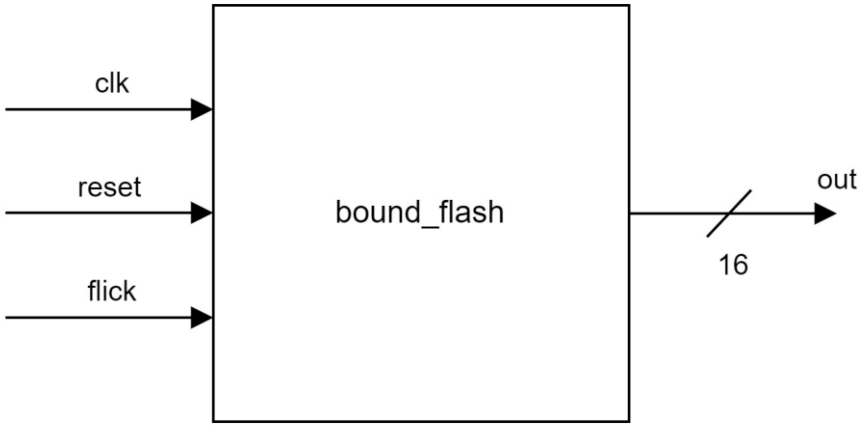


Figure 1: the figure of Bound Flasher System

Signal	Width	In/Out	Description
clk	1	In	Tín hiệu xung clock
reset	1	In	Tín hiệu reset (kích cạnh lên)
flick	1	In	Tín hiệu flick
out	16	Out	Tín hiệu output (16 bit tương ứng 16 đèn)

Table 1: Description of signals in Bound Flasher

2. Functional implementation.

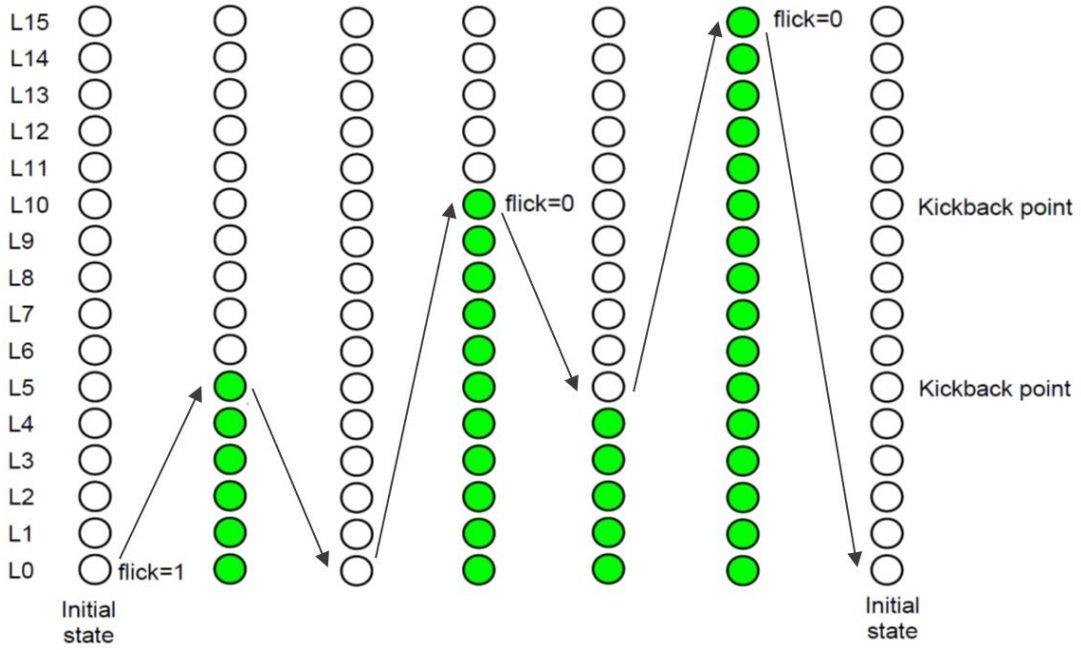
- Implement a 16-bits LEDs system
- System's Operation base on three input signal
  - Reset
  - Clock
  - Flick
- The system specification
  - Clock signal is provided for system inspire of function status. The function operate state's transition at positive edge of the clock signal.
  - Reset signal:
    - HIGH-ACTIVE Reset = 1: System is restarted to Initial State.
    - LOW-ACTIVE Reset = 0: System is started with initial state.
- Flick signal: special input for controlling state transfer.
- At the initial state, all lamps are OFF. If flick signal is ACTIVE, the flasher start operating:
  - The lamps are turned ON gradually from LEDs [0] to LEDs [5].
  - The LEDSs are turned OFF gradually from LEDs [5] to LEDs [0].
  - The LEDSs are turned ON gradually from LEDs [0] to LEDs [10].
  - The LEDSs are turned OFF gradually from LEDs [10] to LEDs [5].
  - The LEDSs are turned ON gradually from LEDs [5] to LEDs [15].
  - Finally, the LEDs s are turned OFF gradually from LEDSs [15] to LEDSs [0], return to initial state.

  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.

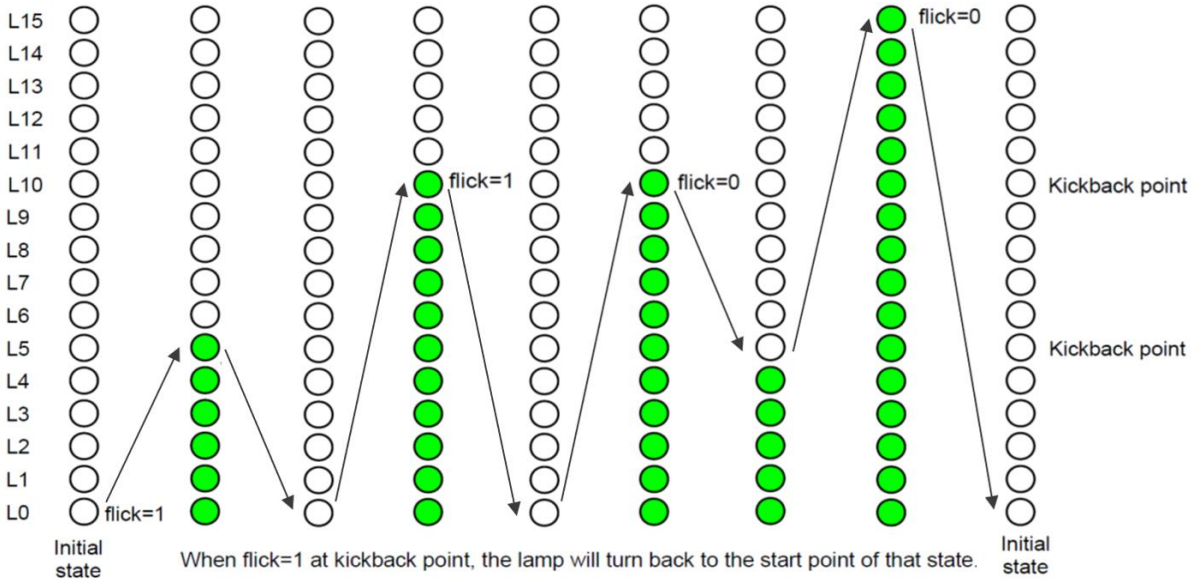
  For simple, kickback point is considered only when the lamps are turned ON gradually, except the first state.

– Some insulations:

- When flick = 0 at kickback points



- When flick = 1 at kickback points (lamp[10])



3. Internal implementation.

3.1. Overall.

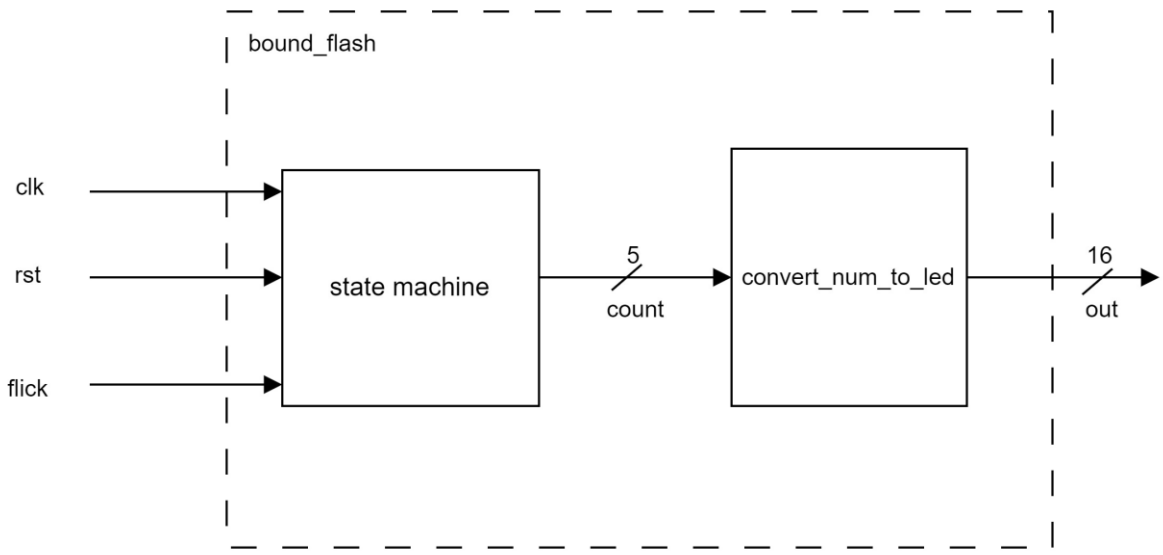


Figure 3.1: Block diagram of Bound Flasher

Block diagram of Bound Flasher Description

Khối state machine: Có 6 state chính từ S0 đến S5. State machine chứa biến state và count sẽ được cập nhật theo chu kì clock. Reset sẽ là tín hiệu bất đồng bộ và flick là tín hiệu đồng bộ theo xung clock.

Khối convert_num_to_led: lấy input là biến count 5-bit để cho ra output 16-bit tương ứng với 16 đèn.

3.2. State Machine

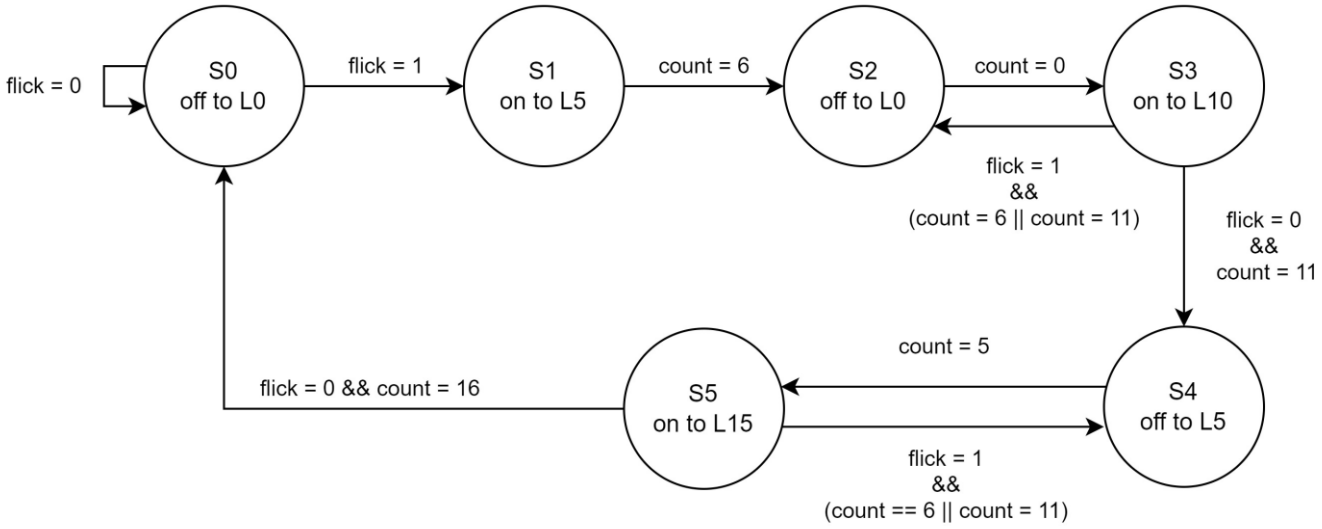


Figure 3.2: State Machine of Bound Flasher

Name	Description
flick	Tín hiệu input flick
count	Số đèn đang bật
L0, L1, … L15	Theo thứ tự đèn 0,1.., 15

Table 3.2: variable name of State machine

Name	Description
S0	Trạng thái bắt đầu, đèn tắt dần xuống L0
S1	Trạng thái đèn bật dần lên L5
S2	Trạng thái đèn tắt dần xuống L0
S3	Trạng thái đèn bật dần đến L10
S4	Trạng thái đèn tắt dần xuống L5
S5	Trạng thái đèn bật dần đến L15

Table 3.3: state name of State machine

4. History

Date	Author	Modified part	Description
2023/03/19	All	All	Mỗi bạn tự code riêng
2023/03/28	All	All	Nhóm họp offline review code và vẽ state machine
2023/3/2	All	All	Nhóm họp offline sửa lỗi khi simulate và hoàn thiện báo cáo

VERILOG CODE:

```
module bound_flash(clk, reset, flick, state, out);
    parameter S0=0, S1=1, S2=2, S3=3, S4=4, S5=5, S6=6;
    input clk, reset, flick;
    output wire [15:0] out;
    output reg [3:0] state;
    reg [4:0] count = 0;

    convert_num_to_led func(.num (count), .led (out));

    always @ (posedge clk or posedge reset) begin
        if (reset) state = S0;
        else begin
            case (state)
                S0:
                    begin
                        if (count>0 )count = count - 1;
                        else begin
                            count = 0;
                            state = (flick == 1)?S1:S0;
                        end
                    end
                S1:
                    begin
                        count = count + 1;
                        if (count==6) state = S2;
                    end
                S2:
                    begin
                        count= count - 1;
                        if (count==0) state = S3;
                    end
                S3:
                    begin
                        count= count + 1;
                        if (count==6) state = (flick==1)?S2:S3;
                        else if (count==11) state = (flick==1)?S2:S4;
                    end
                S4:
                    begin
                        count = count - 1;
                        if (count==5) state = S5;
                    end
                S5:
                    begin
                        count = count + 1;
```

```
                        if (count==6) state = (flick==1)?S4:S5;
                        else if (count==11) state = (flick==1)?S4:S5;
                        else if (count==16) state = S0;
                    end
                default: state = 0;
            endcase
        end
    end
endmodule
```

```
module convert_num_to_led(num,led);
    input [4:0]num;
    output reg[15:0]led;
    reg [4:0]i;
    reg [4:0]num_temp;

    always@(num) begin
        led=16'b0;
        num_temp=num;
        for(i=16; i>0;i=i-1) begin
            if(num_temp>0) begin
                num_temp=num_temp-1;
                led=led << 1;
                led=led | 1'b1;
            end
        end
    end
endmodule
```