CS 2104 02 Hardware Design and Labs 2018 Lab 6
學號：105072123　　姓名：黃海茵

1. 實作過程

(1) 因為 LED 燈是按鍵就亮，但依照 clk/2^26 來熄滅的，所以我把 LED 四個一組實作，以下說明。

```
always @(posedge clk) begin
    clk_count <= (judge1 || judge2 || judge3 || clk_count == 26'd67108863)? 26'd0 : clk_count + 1'b1;
    LED[3:0] <= LED[3:0];
    if (rst_deb) begin
        LED[3:0] <= 4'b0;
        clk_count <= 26'd0;
    end
    else begin
        if (been_ready && key_down[last_change] == 1'b1 && key_num == 4'b0001) LED[3:0] <= {LED[2:0], 1'b1};
        else if (judge1 || clk_count == 26'd67108863 && state == STAY && num0 == 4'd1) LED[3:0] <= LED[3:0] >> 1'b1;
        else LED[3:0] <= LED[3:0];
    end
end

always @(posedge clk) begin
    LED[7:4] <= LED[7:4];
    if (rst_deb) LED[7:4] <= 4'b0;
    else begin
        if (been_ready && key_down[last_change] == 1'b1 && key_num == 4'b0010) LED[7:4] <= {LED[6:4], 1'b1};
        else if (judge2 || clk_count == 26'd67108863 && state == STAY && num0 == 4'd2) LED[7:4] <= LED[7:4] >> 1'b1;
        else LED[7:4] <= LED[7:4];
    end
end

always @(posedge clk) begin
    LED[11:8] <= LED[11:8];
    if (rst_deb) LED[11:8] <= 4'b0;
    else begin
        if (been_ready && key_down[last_change] == 1'b1 && key_num == 4'b0011) LED[11:8] <= {LED[10:8], 1'b1};
        else if (judge3 || clk_count == 26'd67108863 && state == STAY && num0 == 4'd3) LED[11:8] <= LED[11:8] >> 1'b1;
        else LED[11:8] <= LED[11:8];
    end
end
```

judge1 是當電梯樓層下降到 1 樓時，立刻從 0 變成 1 的變數，judge2 和 judge3 也是做相同的事。所以當電梯一降到 1 樓時，LED[3:0]就會馬上熄掉最左邊那個，然後再用 clk_count 開始數到 2^26-1 時，熄掉下一個，2 樓和 3 樓也是用相同的做法。

(2) 這次我用了三個 state 來實作，分別是 STAY, UP, DOWN，以下說明。

```
STAY: begin
    next_num0 = num0;
    if (num0 == 4'd1) begin
        flag =1'b1;
        next_num1 = (LED[11:4] != 8'b0)? 4'd4 : 4'd0;
        next_state = (LED[11:4] != 8'b0 && LED[3:0] == 4'b0)? UP : STAY;
    end
    else if (num0 == 4'd2) begin
        next_num1 = (LED[11:8] != 4'b0 && LED[3:0] == 4'b0 ||
                     LED[11:8] != 4'b0 && LED[3:0] != 4'b0 && flag)? 4'd4 :
                    (LED[11:8] == 4'b0 && LED[3:0] != 4'b0 ||
                     LED[11:8] != 4'b0 && LED[3:0] != 4'b0 && !flag)? 4'd8 : 4'd0;
        next_state = (LED[7:4] != 4'b0 || LED[11:8] == 4'b0 && LED[3:0] == 4'b0)? STAY :
                     (LED[11:8] != 4'b0 && LED[3:0] == 4'b0 ||
                      LED[11:8] != 4'b0 && LED[3:0] != 4'b0 && flag)? UP : DOWN;
    end
    else if (num0 == 4'd3) begin
        flag = 1'b0;
        next_num1 = (LED[7:0] != 8'b0)? 4'd8 : 4'd0;
        next_state = (LED[7:0] != 8'b0 && LED[11:8] == 4'b0)? DOWN : STAY;
    end
end
```

num0 是代表樓層的變數，STAY 的樓層不會改變，所以 next_num0 = num0。而 num1 是代表電梯方向的變數。
當電梯在 1 樓時，我把 flag 拉為 1（在 2 樓時會用到），然後用 LED 燈判斷樓上有人時 next_num1 顯示向上，樓上有人且 1 樓沒人時 next_state 即為 UP，反之即為 STAY。
當電梯在 2 樓時，一樣用 LED 燈判斷 3 樓或 1 樓有沒有人。如果 3 樓有人 1 樓沒人，則 next_num1 就會向上，3 樓沒人 1 樓有人則反之。但若 3 樓和 1 樓同時都有人，則判斷 flag 是否為 1，若 flag 為 1 表示電梯是從 1 樓上來的，則會向上，且 2 樓沒人時 next_state 即為 UP，反之為 DOWN。但若 2 樓還有人，或 1, 3 樓都沒人即為 STAY。
當電梯在 3 樓時，把 flag 降為 0（在 2 樓時會用到），後用 LED 燈判斷樓下有人時 next_num1 顯示向下，樓下有人且 3 樓沒人時，next_state 即為 DOWN，反之即為 STAY。

```
UP: begin
    next_count = (count == 2'd3)? 2'd0 : count + 1'b1;
    next_num0 = (num1 == 4'd7 && num0 != 4'd3)? num0 + 1'b1 : num0;
    next_num1 = (count == 2'd0)? 4'd4 : (num1 == 4'd4)? 4'd5 : (num1 == 4'd5)? 4'd6 : 4'd7;
    next_state = (num0 == 4'd1 && num1 == 4'd7 && LED[7:4] != 4'b0 || num0 == 4'd2 && num1 == 4'd7)? STAY : UP;
    next_judge3 = (num0 == 4'd2 && num1 == 4'd7)? 1'b1 : 1'b0;
    next_judge2 = (num0 == 4'd1 && num1 == 4'd7 && LED[7:4] != 4'b0)? 1'b1 : 1'b0;
end
```

count 是用來跑上升、下降圖示的判斷變數。
在 UP 時，當上升跑到最後一個圖示了，樓層數（next_num0）加 1，反之則維持原數。num1 則依序跑那 4 個圖示。而 jugde3 在準備升到 3 樓時（現在在 2 樓）拉為 1，若是升到 2 樓的話（現在在 1 樓）則是 judge2 拉為 1。
如果現在在 1 樓，上升圖示也跑到最後一個，2 樓又沒人的話，就會繼續上升到 3 樓，所以 next_state 為 UP，反之為 STAY。

```
DOWN: begin
    next_count = (count == 2'd3)? 2'd0 : count + 1'b1;
    next_num0 = (num1 == 4'd11 && num0 != 4'd1)? num0 - 1'b1 : num0;
    next_num1 = (count == 2'd0)? 4'd8 : (num1 == 4'd8)? 4'd9 : (num1 == 4'd9)? 4'd10 : 4'd11;
    next_state = (num0 == 4'd3 && num1 == 4'd11 && LED[7:4] != 4'b0 || num0 == 4'd2 && num1 == 4'd11)? STAY : DOWN;
    next_judge1 = (num0 == 4'd2 && num1 == 4'd11)? 1'b1 : 1'b0;
    next_judge2 = (num0 == 4'd3 && num1 == 4'd11 && LED[7:4] != 4'b0)? 1'b1 : 1'b0;
end
```

DOWN 和 UP 做的事其實很像，只是往另一個方向進行。
在 DOWN 時，當下降跑到最後一個圖示了，樓層數（next_num0）減 1，反之則維持原數。num1 則依序跑那 4 個圖示。而 judge1 在準備降到 1 樓時（現在在 2 樓）拉為 1，若是降到 2 樓的話（現在在 2 樓）則是 judge3 拉為 1。
如果現在在 3 樓，下降圖示也跑到最後一個，2 樓又沒人的話，就會繼續下降到 1 樓，所以 next_state 為 DOWN，反之為 STAY。

2. **學到的東西與遇到的困難**

一開始我把 LED[11:0]全部打在同一個 always block 裡，結果發現按鍵盤時，會干擾到正在熄滅的 LED 燈（會讓它無法熄滅，或是熄滅超級快）。後來把三個樓層的 LED 燈分開就解決這個問題了。

3. **想對老師或助教說的話**

我覺得這次 Lab 的作業說明不太清楚，看了 Demo 影片很多次，還有討論區同學們的發問，才找到全部的規則。例如電梯是否在一到那個樓層時馬上就會熄掉一個 LED 燈，還是到了那個樓層過 clk/2^26 才會熄掉第一個等等的問題……。

老師、助教們辛苦了 ☺☺☺