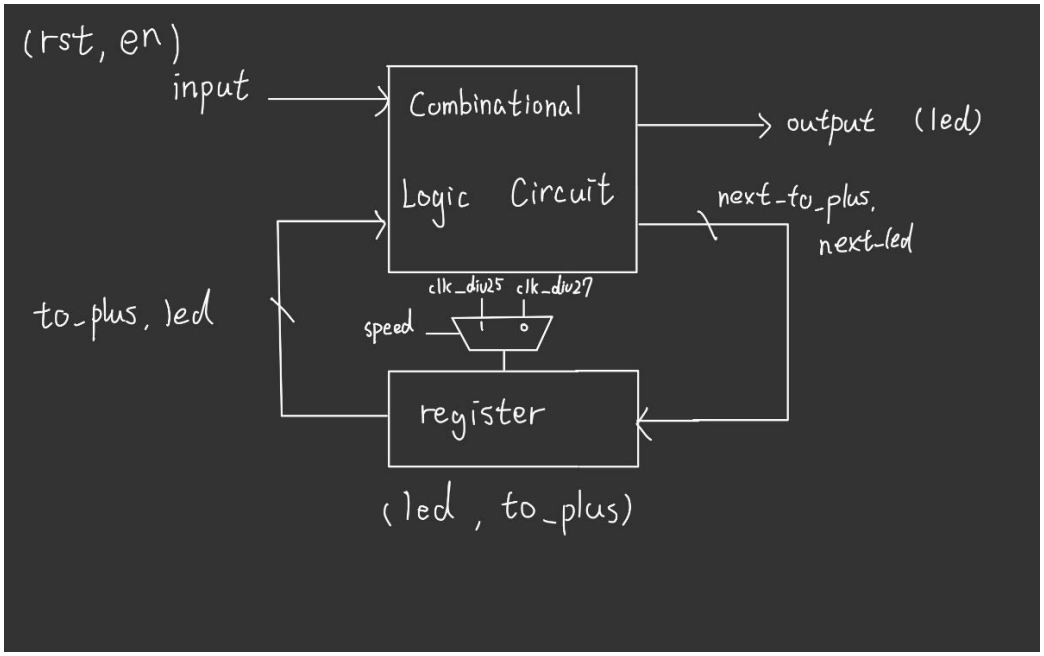
Lab 3

A. Lab Implementation

1.Block diagram

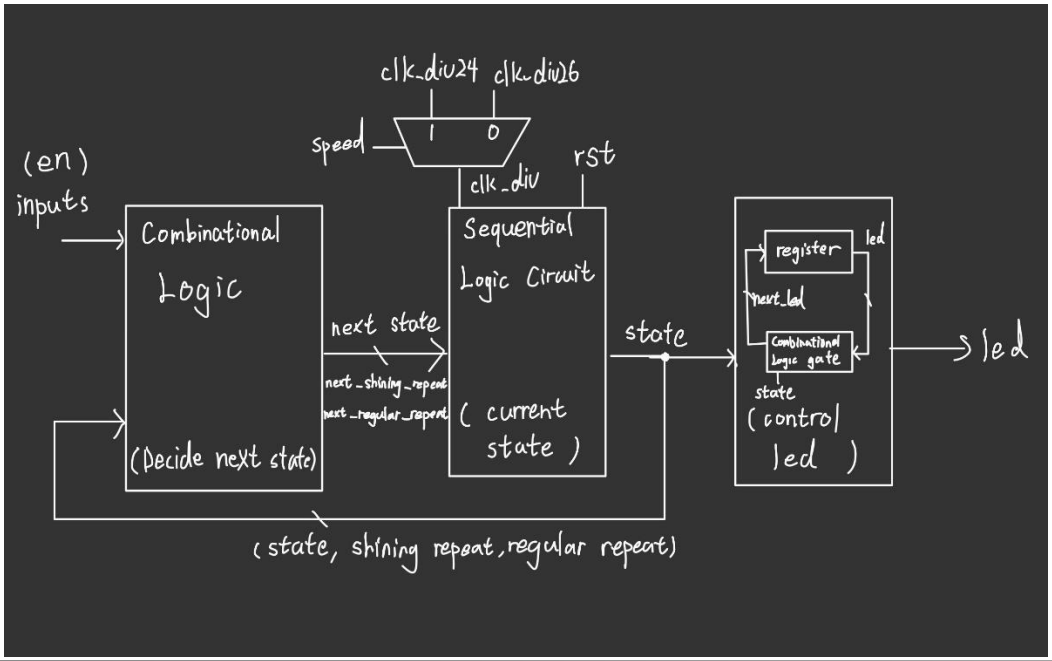
a.Lab3_1:

第一題我用了一個叫 to_plus 的 reg 作為每次 led 要變成下一個狀態所要加上的數值，並在 led 改變後，將 to_plus 向右移一格，當 to_plus = 16b'0001000100010001 時，將他設為 16'b1000100010001000 ，並重複這個循環，而 led 也在等於 16'b1111111111111111 後重置為 0。



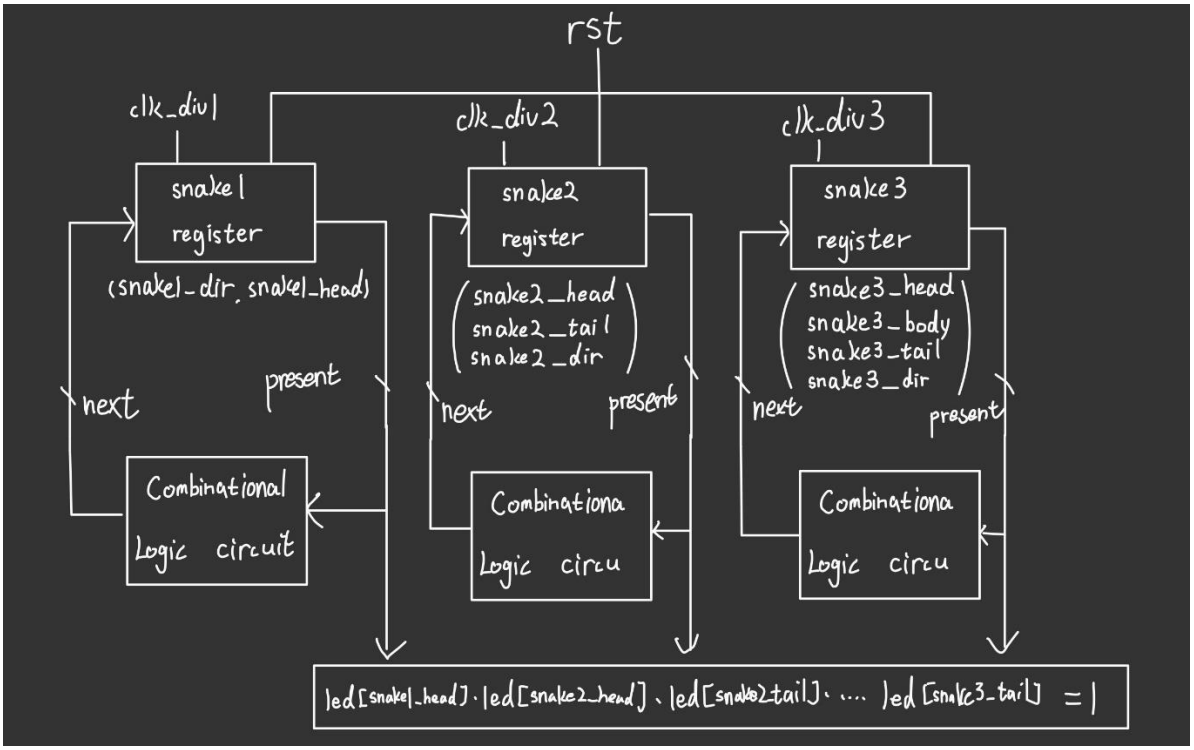
b. Lab3_2:

第二題我用了 3 個狀態，分別為 regular mode，escape mode 和 shining mode，並設了 regular_repeat 和 shining_repeat 來計算分別在兩個 mode 中 led 所重複的次數，並讓各個 mode 在達到切換條件時切換。至於燈泡改變的規律則由當下的 mode 所決定。



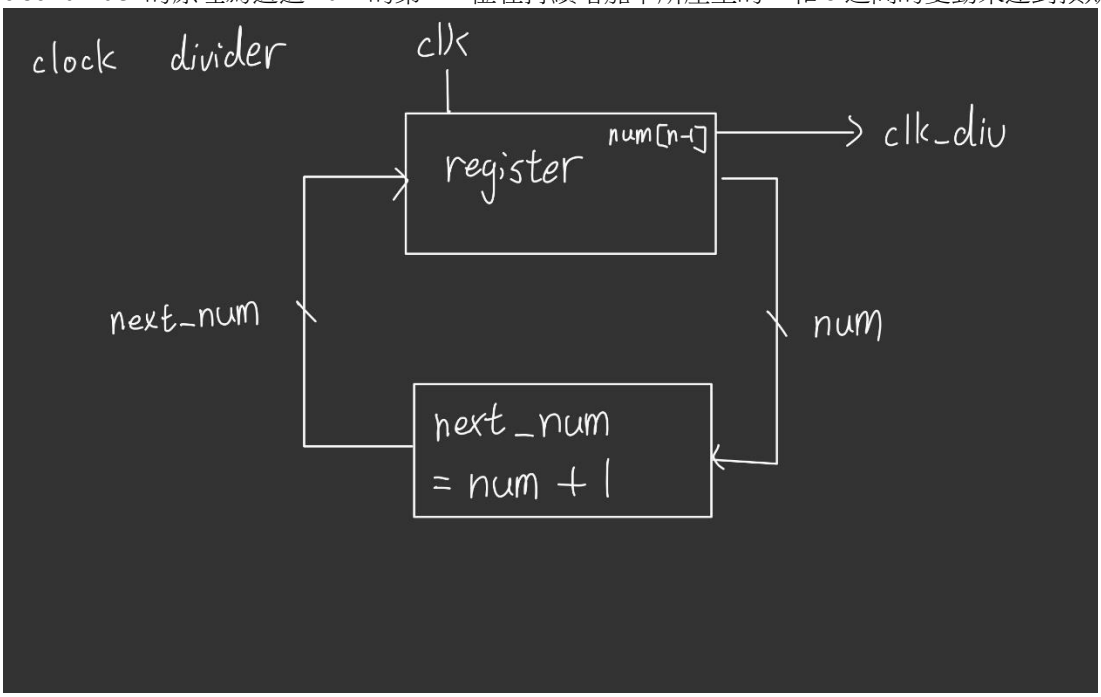
c. Lab3_3:

先用了三個 clock divider 產生的 clk_div 來分別作為三隻蛇的 clk，以達到三隻蛇移動速度不同的效果，並用多個變數，如:snake1_head、snake2_tail 之類的變數來記錄三隻蛇分別的位置，最後再用記錄蛇所在的位置的變數來控制 led 的明暗分布。



d. clock divider

clock divider 的原理為透過 num 的第 n-1 位在持續增加下所產生的 1 和 0 之間的變動來達到預期的效果



2.partial code

a.Lab3_1:

to_plus 的部分我先令了兩個變數，to_plus 和 next_to_plus，next_to_plus 用來儲存下一個 clk edge 的 to_plus，並在下一個 clk edge 賦值給 to_plus。

```
25    wire clk_div,clk_div25,clk_div27;
26    wire[15:0]to_plus;
27    reg [15:0]next_to_plus,next_led;
```

下圖為決定 next_to_plus 的部分

```
51    always@(posedge clk_div , posedge rst)begin
52        if(rst | to_plus == 16'b0000000000000000) begin
53            next_to_plus <= 16'b1000100010001000;
54        end
55        else if (en == 1'b0)begin
56            next_to_plus <= to_plus ;
57        end
58        else if (to_plus == 16'b0001000100010001)begin
59            next_to_plus <= 16'b0000000000000000;
60        end
61        else begin
62            next_to_plus <= (to_plus / 16'd2);
63        end
64    end
65    assign to_plus = next_to_plus;
```

另一個部分為 clk divider 的部分，用 num 第 n-1 位 0 和 1 之間的變動來達到題目要的效果

```
2   module clock_divider(clk,clk_div);
3       input clk;
4       output clk_div;
5       parameter n = 25;
6       reg[n-1:0]num;
7       wire[n-1:0]next_num;
8       always@(posedge clk)begin
9           num <= next_num;
10      end
11      assign next_num = num + 1;
12      assign clk_div = num[n-1];
13  endmodule
```

下圖為速度控制的部分，利用 speed 是否為 1 來決定 clk_div 是速度較快的 clk_div25 還是較慢的 clk_div27。

```
33    wire clk_div,clk_div25,clk_div27;
34    assign clk_div = (speed)?clk_div25 : clk_div27;
```

最後則是控制 led 的部分，用 to_plus 和當前狀態的 led 組成 next_led

```
37  always@(posedge clk_div , posedge rst)begin
38      if(rst) begin
39          next_led <= 16'b0000000000000000;//reset the led
40      end
41      else if (en == 1'b0)begin
42          next_led <= led ;
43      end
44      else if (led == 16'b1111111111111111)begin
45          next_led <= 16'b0000000000000000;
46      end
47      else begin
48          next_led <= led + to_plus;
49      end
50  end
51
52  assign led = next_led;
```

b.Lab3_2:

這部分的 code 分成多個部分，分別為控制 mode、控制 regular_repeat、 控制 shining_repeat 和控制 led 的部分，至於 clk_div 的控制則和 lab3_1 一樣

下圖為控制 mode 的部分和 mode 需要轉變的條件，由於有些需要空置一個 cycle，所以我用了以下的方法來控制，這邊以以 regular mode 為例：將 mode 的改變條件設為該 mode 最後的狀態，在那個 clk edge 時，next_mode 會等於 escape mode，而這樣下個 cycle 就仍然會是 regular mode，因此只要在 regular mode 中多設一個條件，讓 led 在 regular repeat = 3 且 led 為全亮時，next_led 依然為全亮，這樣即可以達到 demo 影片中的效果。

```
always@(posedge clk_div , posedge rst ,posedge change_state)begin
    if(rst)begin
        state <= regular_mode;
    end
    else if (change_state)begin
        state <= next_state;
        change_state <= 0;
    end
    else begin
        state <= next_state;
    end
end

always@(*)begin
    if(!en)
        next_state <= state;
    else if(state == regular_mode & regular_repeat == 2'b11 & dir == 0 & led == 16'b1111111111111111)begin
        next_state <= esscape_mode_down;
    end
    else if(state == regular_mode & regular_repeat == 2'b11 & dir == 1 & led == 16'b1111111111111111)begin
        next_state <= esscape_mode_up;
    end
    else if(state == esscape_mode_down & dir == 1)begin
        next_state <= esscape_mode_up;
        change_state <= 1;
    end
    else if(state == esscape_mode_up & dir == 0)begin
        next_state <= esscape_mode_down;
        change_state <= 1;
    end
    else if(state == esscape_mode_down & led == 16'b0000000000000000)begin
        next_state <= shining_mode;
    end
    else if(state == esscape_mode_up & led == 16'b1111111111111111)begin
        next_state <= regular_mode;
    end
    else if(state == shining_mode & shining_repeat == 3'b100 & led == 16'b1111111111111111)begin
        next_state <= regular_mode;
    end
    else
        next_state <= state;
end
```

下圖為控制 regular_repeate 的部分，regular_repeat 會在每個 clk cylce 加一，並再重複三次後歸 0

```
always@(posedge clk_div , posedge rst)begin
    if(rst)begin
        regular_repeat <= 2'b00;
    end
    else begin
        regular_repeat <= next_regular_repeat;
    end
end

always @(*) begin
    if(!en)
        next_regular_repeat <= regular_repeat;
    else if(regular_repeat == 2'b11)
        next_regular_repeat <= 2'b00;
    else if(state == regular_mode & led == 16'b1110111011101110)
        next_regular_repeat <= regular_repeat + 2'b01;
    else
        next_regular_repeat <= regular_repeat;
end
```

下圖為控制 shining_repeat 的部分，和 regular_repeat 是一樣的概念

```
104   always@(posedge clk_div , posedge rst)begin
105       if(rst)begin
106           shining_repeat <= 3'b000;
107       end
108       else begin
109           shining_repeat <= next_shining_repeat;
110       end
111   end
112
113   always @(*) begin
114       if(!en)
115           next_shining_repeat <= shining_repeat;
116       else if(shining_repeat == 3'b101)
117           next_shining_repeat <= 3'b000;
118       else if(state == shining_mode & led == 16'b1111111111111111)
119           next_shining_repeat <= shining_repeat + 3'b001;
120       else
121           next_shining_repeat <= shining_repeat;
122   end
```

最後則為控制 led 在不同 mode 中所需做的轉換的部分

```
124   always@(posedge clk_div , posedge rst)begin
125       if(rst)begin
126           led <= 16'b0000000000000000;
127       end
128       else if (change_state)begin
129           led <= next_led;
130           change_state <= 0;
131       end
132       else begin
133           led <= next_led;
134       end
135   end
136   always @(*)begin
137       if(!en)
138           next_led <= led;
139       else if(state == regular_mode )begin
140           if(led == 16'b0000000000000000)
141               next_led <= 16'b1000100010001000;
142           else if(led == 16'b1000100010001000)
143               next_led <= 16'b1100110011001100;
144           else if(led == 16'b1100110011001100)
145               next_led <= 16'b1110111011101110;
146           else if(led == 16'b1110111011101110)
147               next_led <= 16'b1111111111111111;
148           else if(led == 16'b1111111111111111 & regular_repeat == 2'b11)
149               next_led <= 16'b1111111111111111;
150           else if(led == 16'b1111111111111111)
151               next_led <= 16'b0000000000000000;
152       end
153       else if(state == esscape_mode_down)begin
154           if(dir)
155               next_led <= led * 4 + 3;
156           else
157               next_led <= led >> 2;
158       end
159       else if(state == esscape_mode_up)begin
160           if(dir)
161               next_led <= led * 4 + 3;
162           else
163               next_led <= led >> 2;
164       end
165       else if(state == shining_mode)begin
166           if(led == 16'b1111111111111111)
167               next_led <= 16'b0000000000000000;
168           else if(led == 16'b0000000000000000 && shining_repeat == 3'b101)
169               next_led <= 16'b1000100010001000;
170           else if(led == 16'b0000000000000000)
171               next_led <= 16'b1111111111111111;
172       end
173       else
174           next_led <= led;
175   end
```

c. Lab3_3:

第三題分為兩大部分，分別為 led 的控制和蛇的位置以及下一步行動的判斷

下圖為 cik_div 的部分，我使用了三個 clock divider 來控制三隻蛇分別的移動頻率。

```
26    clock_divider #(24) div24(.clk(clk), .clk_div(clk_div1));
27    clock_divider #(25) div25(.clk(clk), .clk_div(clk_div2));
28    clock_divider #(26) div26(.clk(clk), .clk_div(clk_div3));
```

```
always @(posedge clk_div1,posedge rst)begin
    if(rst)begin
        snake1_head <= 4'b1111;
        snake1_dir <= RIGHT;
    end
    else if (en) begin
        snake1_head <= next_snake1_head;
        snake1_dir <= next_snake1_dir;
    end
end
```

(snake2、3 也和左圖一樣)

三隻蛇的下一步則分別使用三個 always block 來判斷，看蛇的左右是否有其他蛇或是牆壁，如下圖所示(以 snake1 作為例子

```
82  always @(*) begin
83      if((snake1_head - 4'b0001 == snake2_head |  snake1_head - 4'b0001 == snake2_tail )& snake1_dir == RIGHT) begin
84          if(snake1_head == 4'b1111)begin
85              next_snake1_dir <= RIGHT;
86              next_snake1_head <= snake1_head;
87          end
88          else begin
89              next_snake1_dir <= LEFT;
90              next_snake1_head <= snake1_head + 4'b0001;
91          end
92      end
93      else if(snake1_dir == LEFT & snake1_head == 4'b1111)begin
94          if(snake1_head - 4'b0001 == snake2_head |  snake1_head - 4'b0001 == snake2_tail )begin
95              next_snake1_dir <= RIGHT;
96              next_snake1_head <= snake1_head;
97          end
98          else begin
99              next_snake1_dir <=RIGHT;
100             next_snake1_head <= snake1_head - 4'b0001;
101         end
102     end
103     else if(snake1_dir == LEFT)begin
104         next_snake1_dir <= LEFT;
105         next_snake1_head <= snake1_head + 4'b0001;
106     end
107     else begin
108         next_snake1_dir <= RIGHT;
109         next_snake1_head <= snake1_head - 4'b0001;
110     end
111 end
```

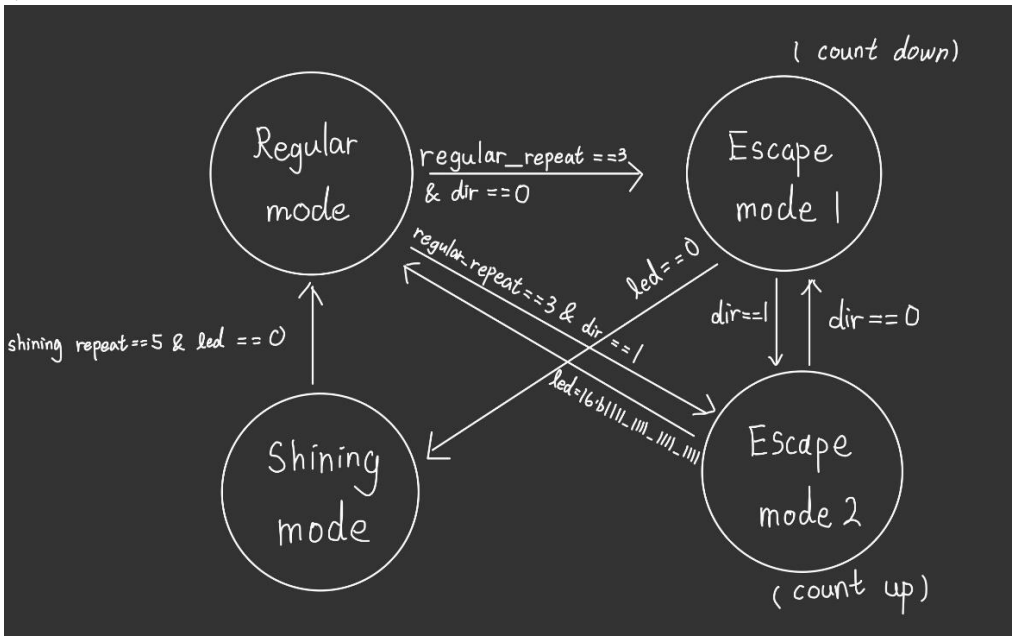
下圖則是控制 led 的部分

```
213   always@(*) begin
214       led = 16'b0000000000000000;
215       led[snake1_head] = 1'b1;
216       led[snake2_head] = 1'b1;
217       led[snake2_tail] = 1'b1;
218       led[snake3_head] = 1'b1;
219       led[snake3_body] = 1'b1;
220       led[snake3_tail] = 1'b1;
221   end
222
```

4.FSM

a.Lab2_2:

下圖為 FSM 圖。



Questions and Discussions

A. In lab3_1, rst has the highest priority than any other signal (in most hardware designs, it's true as well). How do you implement that?

Ans:

在所有賦值的 always block 中的最前面加上 if(rst)的條件判斷式，並在 always @後面的的括號中加入 posedge rst，如下圖所示，就能達到題目所要求的目的。

```
37    always@(posedge clk_div , posedge rst)begin
38        if(rst) begin
39            next_led <= 16'b0000000000000000;//reset the led
40        end
41        else if (en == 1'b0)begin
42            next_led <= led ;
43        end
44        else if (led == 16'b1111111111111111)begin
45            next_led <= 16'b0000000000000000;
46        end
47        else begin
48            next_led <= led + to_plus;
49        end
50    end
51
```

B. In lab3_3, we simplify the direction policies of snakes to only apply at the moment the snake is going to move. What if the policies affect every time snakes meet each other? (Simply explain what you are going to change in your code). Ex: At t cycle of 6Hz clock: when both Snake1 and Snake11 are triggered, they move.

Ans:

將原本三隻蛇分別負責賦值的 always block(下圖)中的 dir <= next_dir 拿出來寫在另一個 always block 中，並將其的 posedge clk_div 都設成移動速度最快的蛇的 clk_div。

```
43  always @(posedge clk_div1,posedge rst)begin
44      if(rst)begin
45          snake1_head <= 4'b1111;
46          snake1_dir <= RIGHT;
47      end
48      else if (en) begin
49          snake1_head <= next_snake1_head;
50          snake1_dir <= next_snake1_dir;
51      end
52  end
```

B. Problem Encountered

a. 寫 lab3-1 時始不知道在修改 code 後需要重新合成，所以 program device 時都一直丟入最原始的資料，造成結果都一樣是錯的，後來問過同學後才發現需要重新合成。

b. 則是卡在有些 clk 需要停一拍再做切換的部分，一開始都會直接轉換過去，後來畫圖之後想了一下，又回去看上次 lab2 的 practice 才想到用下圖的方法就可以延後一個 clk cycle。

c. 3-3 一開始在想的時候卡在要如何讓蛇在相遇時，只讓先動的蛇判斷他的下一步，而原本的維持下來，後來和同學討論後才知道只要用不同的 clk_div 就可以在蛇需要移動時才將他的下一步賦值給他，並在不需要移動時只做判斷就好。

C. Suggestions

有一天小明走著進超商繳完帳單後，坐著輪椅出來
知道為什麼嗎

.
.
.

因為他繳費了(腳廢