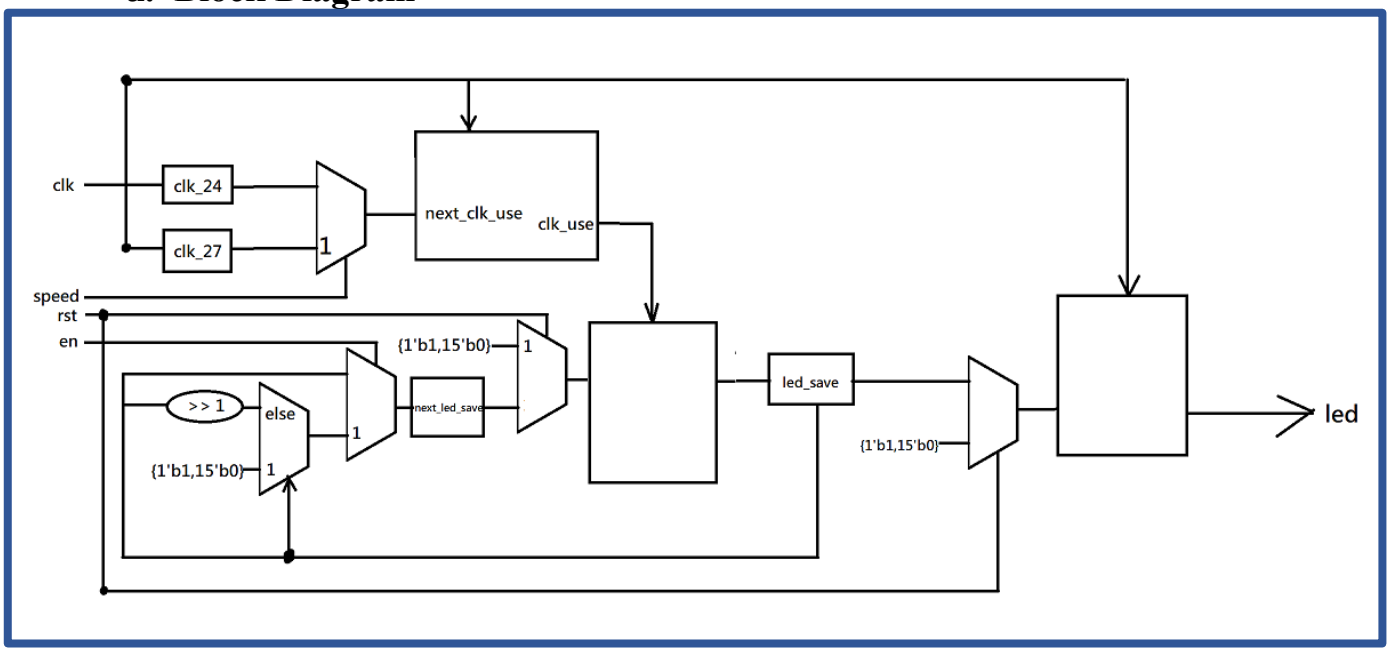
Lab3

學號:109062110 姓名:祝語辰

# A. Lab Implementation

## 1. Lab3_1:

### a. Block Diagram




### b. Kernal code explanation

```
always@(posedge clk)begin
    clk_use <= next_clk_use;
end

always@(*)begin
    if(speed) next_clk_use = clk_27;
    else next_clk_use = clk_24;
end
```

以上是我的 code 中的核心部分。利用一個 flipflop 來做 clock 的更新，

```
always@(posedge clk_use)begin
    if(rst) led_save <= {1'b1,15'b0};
    else led_save <= next_led_save;
end
```

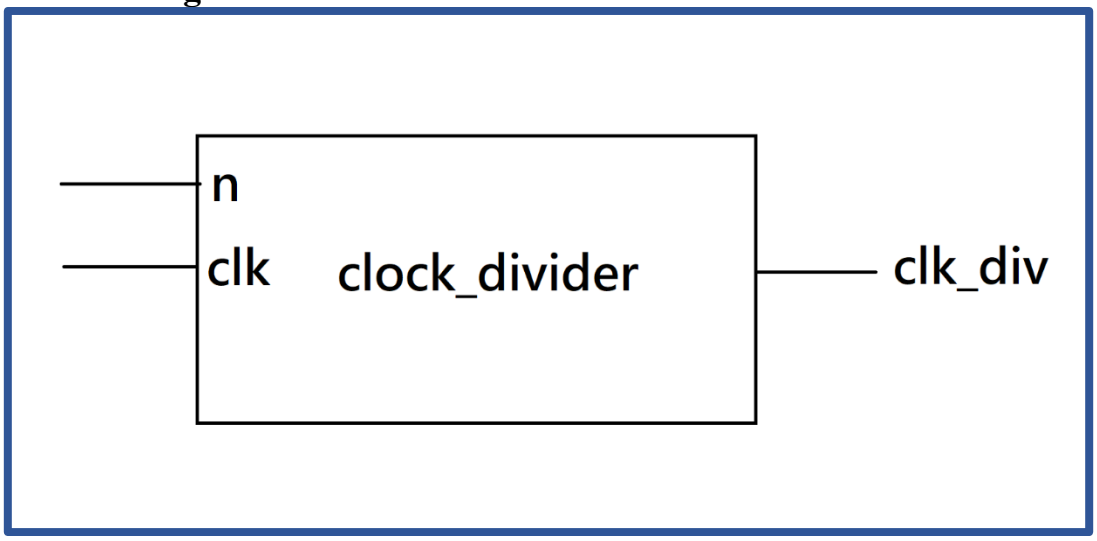
然後用這個 clock 來做 led 的更新。

### c. FSM

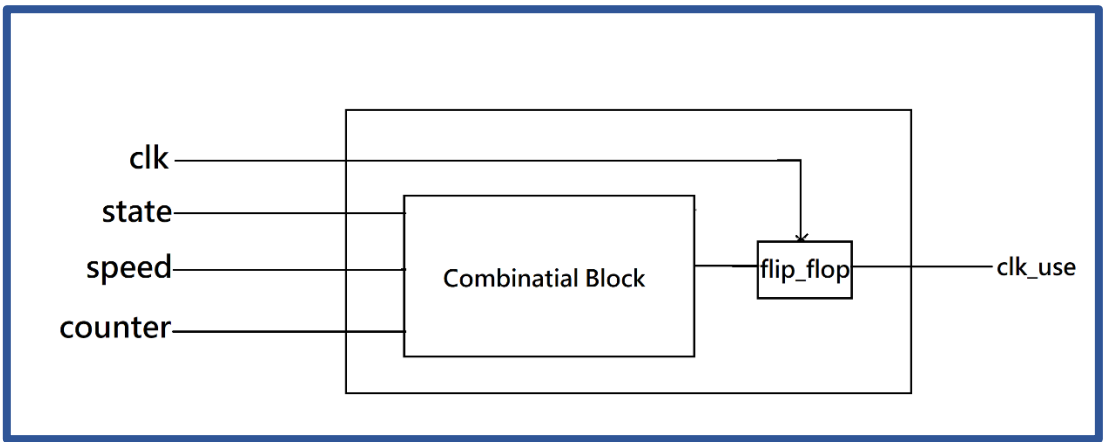
我的 lab3_1 並未使用 FSM。

2. **Lab3_2:**

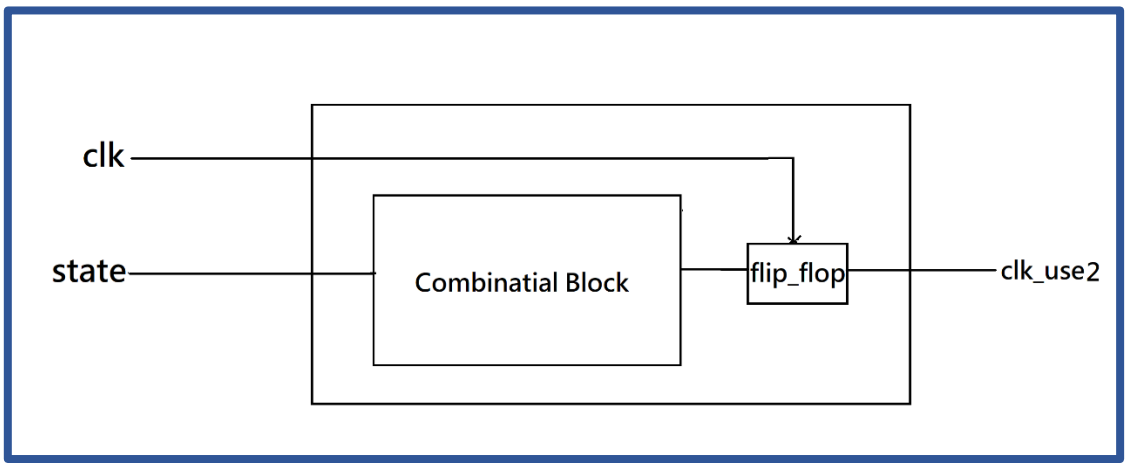
**a. Block Diagram**




上圖為 clock divider，可以會根據 n 除頻，clk_div 的頻率為 clk 的 2n 倍慢。

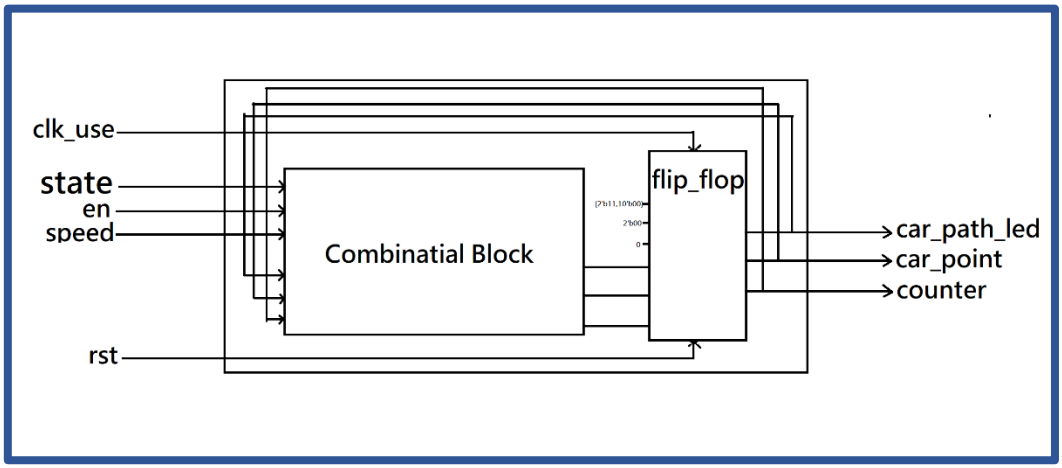



上圖為 clk_use 的 sequential block。透過 combinatial block 來決定 next_clk_use，內部結構請參考 FSM 說明。

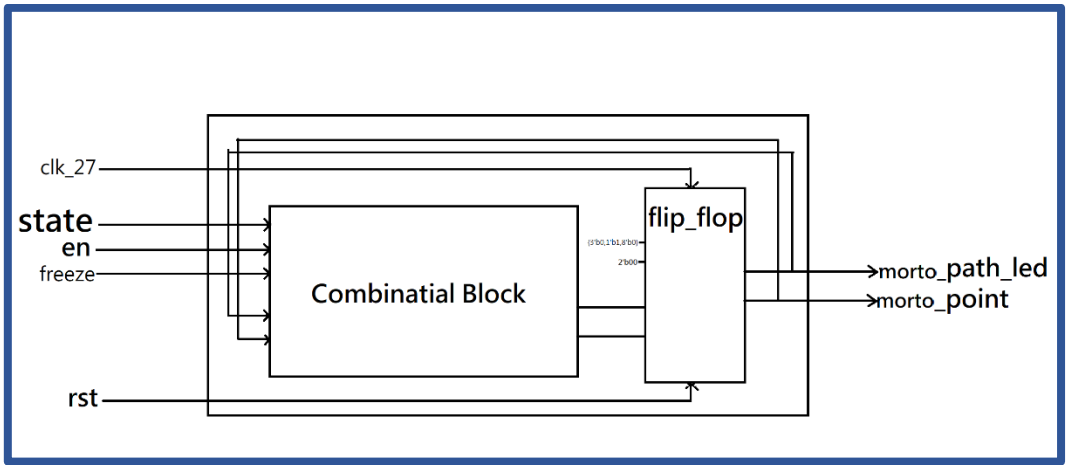



上圖為 clk_use2 的 sequential block。透過 combinatial block

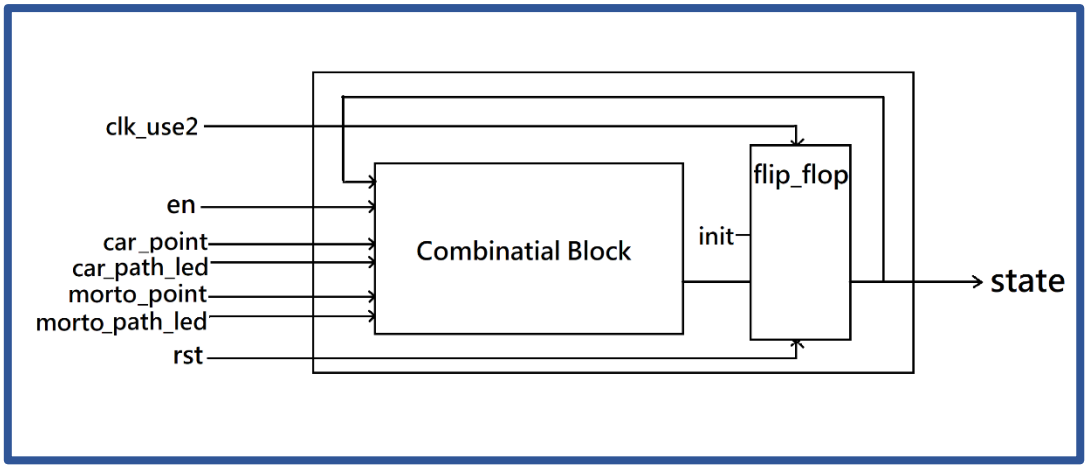
來決定 next_clk_use2，內部結構請參考 FSM 說明。




上圖為 Car 相關訊號的 sequential block。combinatial block 的內部結構請參考 FSM 說明。

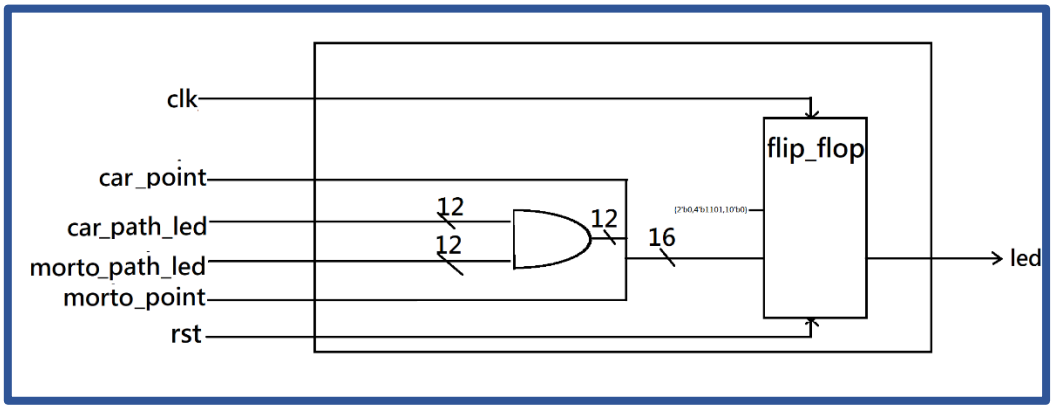



上圖為 motorcycle 相關訊號的 sequential block。combinatial block 的內部結構請參考 FSM 說明。




上圖為 state 相關訊號的 sequential block。combinatial block

的內部結構請參考 FSM 說明。




上圖為最終輸出訊號的 sequential block。

## b. Kernal code explanation

```
114             next_car_path_led = {2'b11,10'b0};
115             next_car_point = 2'b0;
116         end
117         racing:begin
118             if(en) begin
119                 next_car_path_led = car_path_led >> 1;
120             end
121             if (speed) begin
122                 next_counter = counter + 1;
123             end
124             else if(counter!=0)begin
125                 next_counter = 4;
126             end
127         end
128         C_win:begin
```

使用 counter 來記錄有幾個 cycle。

```
always@(posedge clk_use2 or negedge rst)begin//
    if(rst) state <= init;
    else state <= next_state;
end

always@(*)begin
    case(state)
        init:begin
            if(en) next_state = racing;
            else next_state = init;
        end
        racing:begin
            if(morto_path_led == 12'b1 && morto_point == 2'b11) next_state = M_win;
            else if(morto_path_led == 12'b1 && morto_point != 2'b11) next_state = M_finish;
            else if(car_path_led == 12'b11 && car_point != 2'b11) next_state = C_finish;
            else if (car_path_led == 12'b11 && car_point == 2'b11) next_state = C_win;
            else next_state = racing;
        end
        C_win:begin
            next_state = init;
        end
        M_win:begin
            next_state = init;
        end
        C_finish:begin
            next_state = racing;
        end
        M_finish:begin
            next_state = racing;
        end
        default:begin
            next_state = init;
        end
    endcase
end
```

在不同 state，的 clock period 會不同，以此來達到，獲勝獲得分時，需要停頓一定時長的要求。

**c. FSM**

除了在 racing state 利用 car 和 motorcycle 的車燈是否到達終點以及當前的得分數來決定下一 state 的轉換。其他 state 都是提一個 cycle 之後就轉到下一個，至於在各個 state 要停多久(即一個 cycle 到底多長，是會因為各個 state 的不同，而影響到 clk_use2 是要接到 clk_24 或 clk_27。

## B. Questions and Discussions

**1. Robin feels that the freezing weapon in Batmobile is not fair. He upgrades his motorcycle with a shield such that the Motorcycle can only be frozen for three consecutive cycles in a round. How do you modify your racing emulator design?**

```
always@(*)begin
    next_morto_path_led = morto_path_led;
    next_morto_point = morto_point;
    case(state)
        init:begin
            next_morto_path_led = {3'b0,1'b1,8'b0};
            next_morto_point = 2'b0;
        end
        racing:begin
            if(en && freeze !=1 ) next_morto_path_led = morto_path_led >> 1;
        end
        M_win:begin
            next_morto_path_led = 12'b1111_1111_1111;
            next_morto_point = morto_point;
        end
        M_finish:begin
            next_morto_path_led = {3'b0,1'b1,8'b0};
            next_morto_point = morto_point+1;
        end
        C_finish:begin
            next_morto_path_led = {3'b0,1'b1,8'b0};
            next_morto_point = morto_point;
        end
        C_win:begin
            next_morto_path_led = {3'b0,1'b1,8'b0};
            next_morto_point = 2'b0;
        end
        default:begin
            next_morto_path_led = {3'b0,1'b1,8'b0};
            next_morto_point = morto_point;
        end
    endcase
end
```

以上是我的 code 中處理摩托車燈號的部分。如果限制只凍結 3 個 cycle，會使用類似我處理車子加速的方式，使用一個 counter 來計數，並且在 if 的判斷下，多加一個限制，限制該 counter 的值要小於多少。

**2. In lab3_2, if you implemented Method 1, discuss how you can implement Method 2. Or if you implemented Method 2, discuss how you can implement Method 1. Discuss your concept as detailed as possible, and you are encouraged to use block diagrams and FSMs to illustrate your design.**

我在我的作業中，用的是 Method 2，因此在這裡討論用 Method 1。如果要使用 Method 1，又可以使用 counter 來輔助。因為這兩個 clock 之間有整數倍數關係，所以可以使用 counter 來計數。也就是說，使用 100 MHz / 2^24 來做主要的更新的 clock。在需要 100MHz / 2^27 時，就可以用 counter 做控制，從 0~7(2^27 是 2^24 的 8 倍)

## C. Problem Encountered

在 debug 時，有時候 code 看起來沒問題但是跑在板子卻出了問題，這個時候如果能先模擬，用 waveform 來 debug 就會很方便。

一直以來，我都覺得 testbench 有點難寫，會這樣覺得是因為在研究助教寫的

testbench 時，總會覺得寫得好複雜，但這一次實際去試著寫，發現其實沒有那麼複雜，只寫一個 initial 就已經可以滿足我 debug 的需求了。

D. Suggestions

無