Report_dcs193 HW02 蔡東宏 110511277

1. 程式說明:

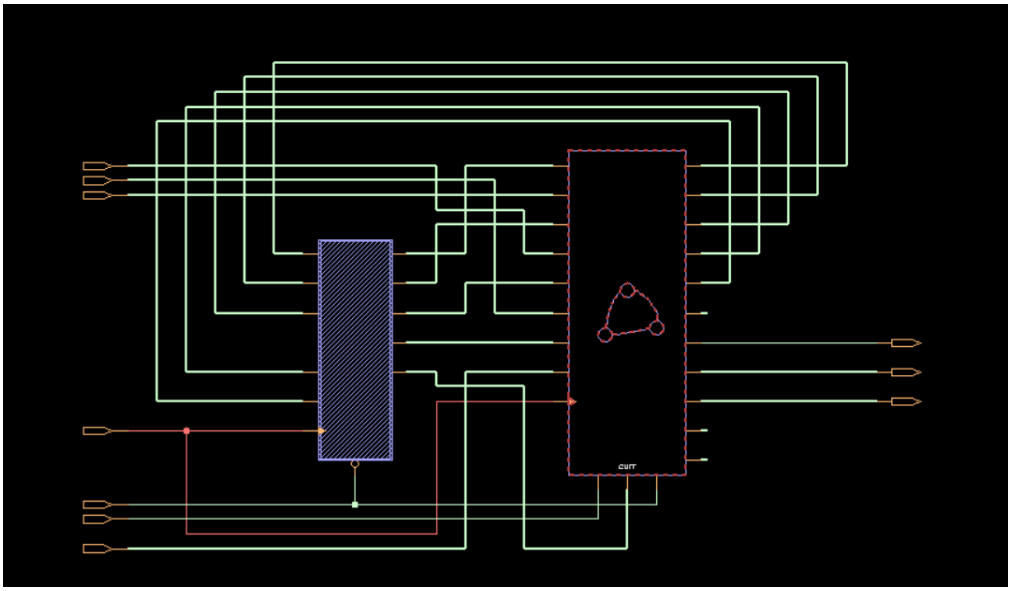
此題為 Traffic Light Controller，輸入為主幹道直走、左轉以及支幹道直走、左轉的車輛，輸出為主幹道及支幹道紅綠燈的變化。此題除了 FSM，我還使用 counter 去紀錄這個 state 跑了幾次，在 state 還沒跑完需要的 cycle 時，都維持一樣的 state，當 count = 需要的 cycle 時，state 才會根據 input 決定要往哪一個 state 走，而在第一次進去某個 state 時，我就會判斷這個 state 需要跑多少 cycle。而我將一組黃紅燈放在一起，當進到其中一組黃紅燈，count = 0 時，我會輸出黃燈，其他則輸出紅燈(而紅燈的次數會根據所在的 state 不同而不同)。

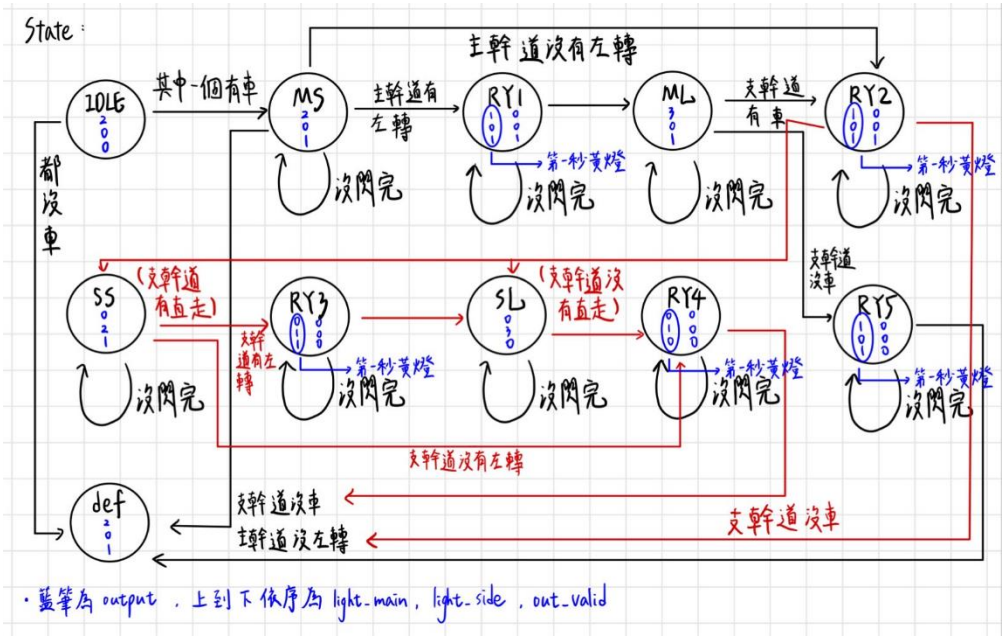
拿主幹道直走舉例:

1. 73~78 行: 當主幹道直走的車大於 4 台，target = 7，所以這個 state 需要跑 8 次才能進到下一個 state。
2. 81~91 行: 當 count 不等於 target 時，下個 state 會維持原本的 state；而當 count 等於 target，再根據 input 的情況判斷下個 state 要去哪裡(要進下一個 state 前須將 count 歸 0)。

```
73      MS:begin
74        if(ms>4)begin
75          target=7;
76        end
77        else
78          target=3;
79        out_valid=1;
80        cnt=cntt+1;
81        if(cntt!=target)
82          next=MS;
83        else begin
84          cnt=0;
85          if(ml!=0)
86            next=RY1;
87          else if(ss!=0||sl!=0)
88            next=RY2;
89          else
90            next=def;
91        end
92      end
```

2. 架構圖:





3. 優化過程:

1. 一開始我的紅黃燈只有寫兩組，分別是主幹道和支幹道，但如此一來，我每次進到這個 state 的時候，我就需要判斷紅燈要閃兩秒或一秒，而且下一個要去的 state 也沒有甚麼共同的特性，這樣變成我還需要用更多的判斷式去執行我的 next_state，因此，最後我寫了 5 個紅黃燈的 state，如此一來，可以大量減少判斷式讓面積變得更小。
2. 透過調整 parameter 的值，讓 state 之間的變化盡可能只需要 1bit，我嘗試調整不同的值，最後讓面積成功有些微的減少。

4. 結論:

這次作業為 traffic light controller，我用了 11 個 state 的 FSM 去解決這次的問題。這次的作業與上次不同的是這次的需要一個 timer 去控制 state 執行的次數，這次的面積沒有大幅度的減少，應該是因為寫法都滿固定的，只有透過增加 state 讓判斷式變小，以及改變 combinational 裡面的一些寫法，讓面積稍為的減少。