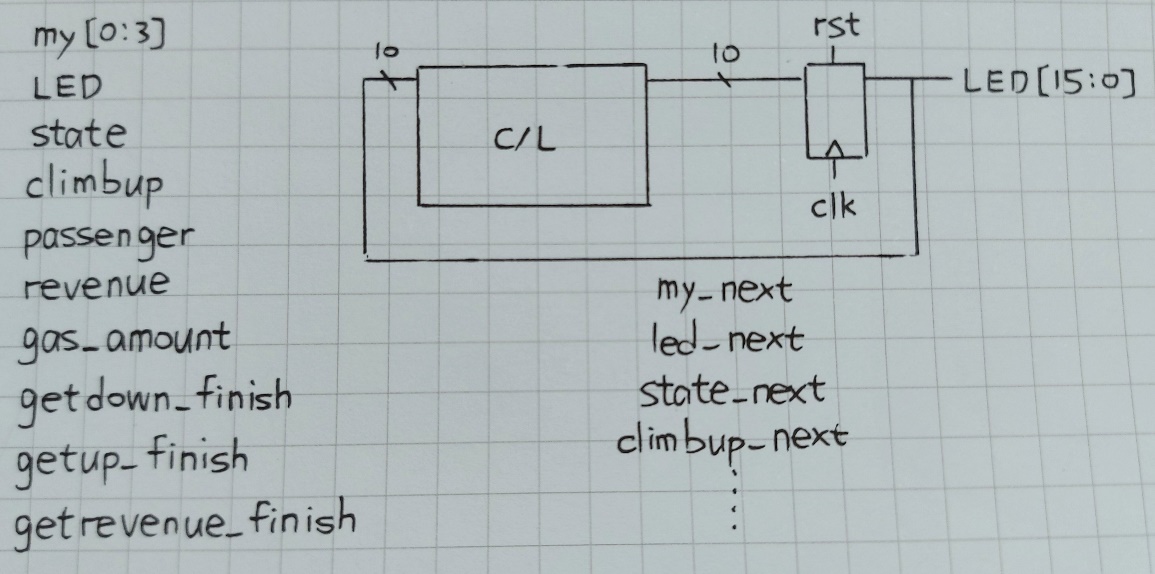
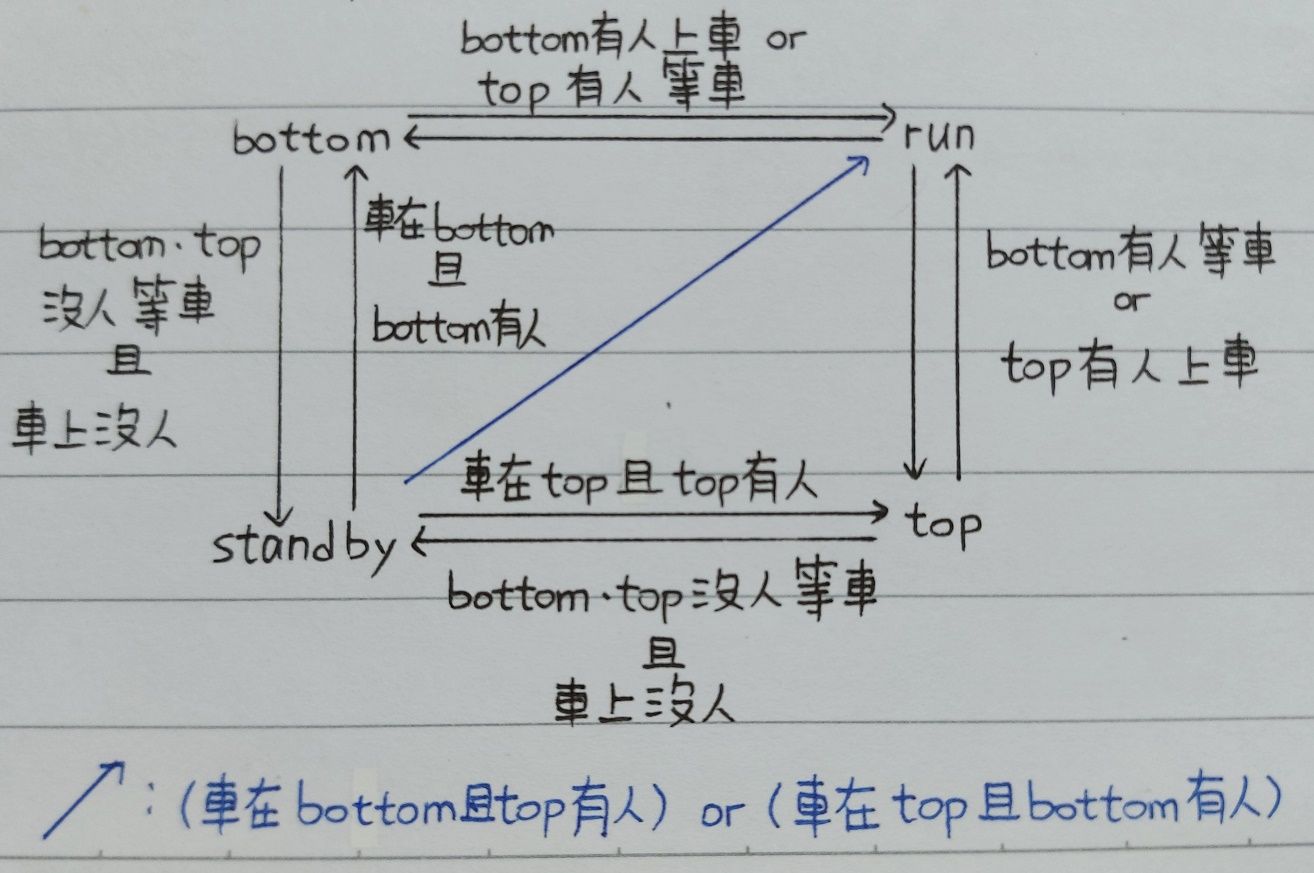
|  |  |
| --- | --- |
| **Lab 6** | |
| 學號: 109062318 | 姓名: 簡弘哲 |

1. 實作過程

block diagram:



State diagram:



我的FSM總共有4個state，分別是BOTTOM(在山下),TOP(在山上),STANDBY(待命),RUN(行駛)。

BOTTOM,TOP處理上下車、付錢、加滿油。RUN根據1 bit變數climbup決定目前是往上還是往下。STANDBY維持所有的變數以及檢查兩個公車站有沒有人在等車。

BOTTOM,TOP:

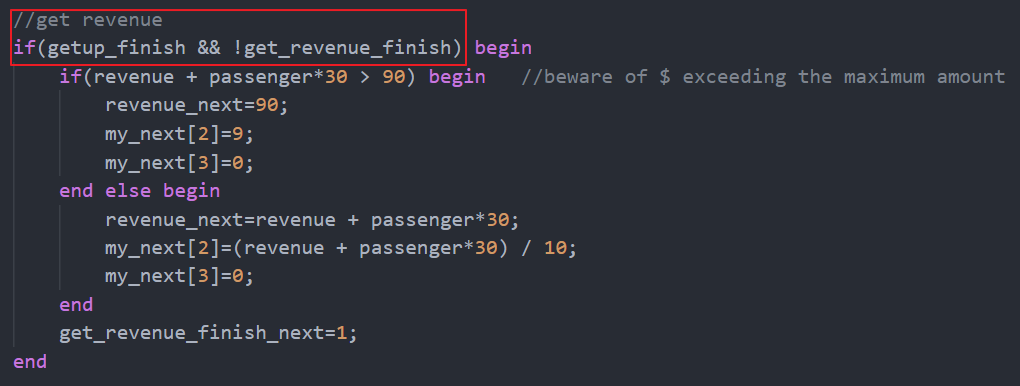
(圖1:在BOTTOM中處理上下車)

一張含有 文字 的圖片

自動產生的描述

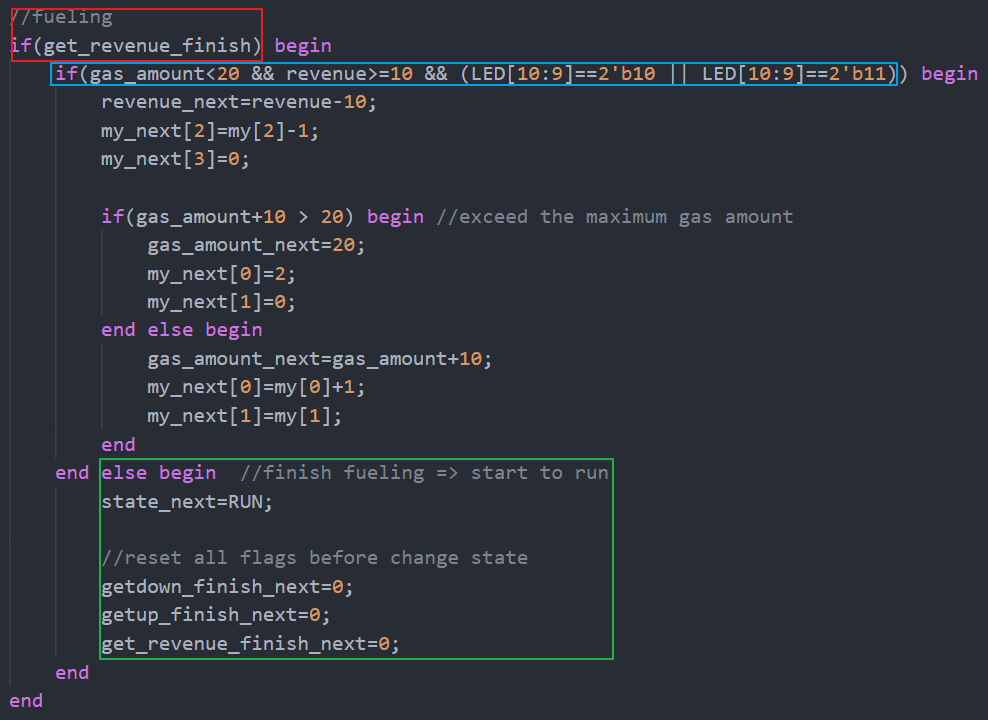
紅框部分是檢查上下車之前各個flag有沒有達到預期的值，例如下車前要先判斷乘客數量>0與是否已完成下車，下車用{}符號去實作left shift。值得注意的地方是藍框部分，如果已經完成下車而且還沒上車的時候，需要檢查有沒有人在等車，如果都沒有人在等就切換到STANDBY待命，直到新的人出現。完成上車後記得將getup\_finish flag拉起。

(圖2:在BOTTOM中處理收錢)



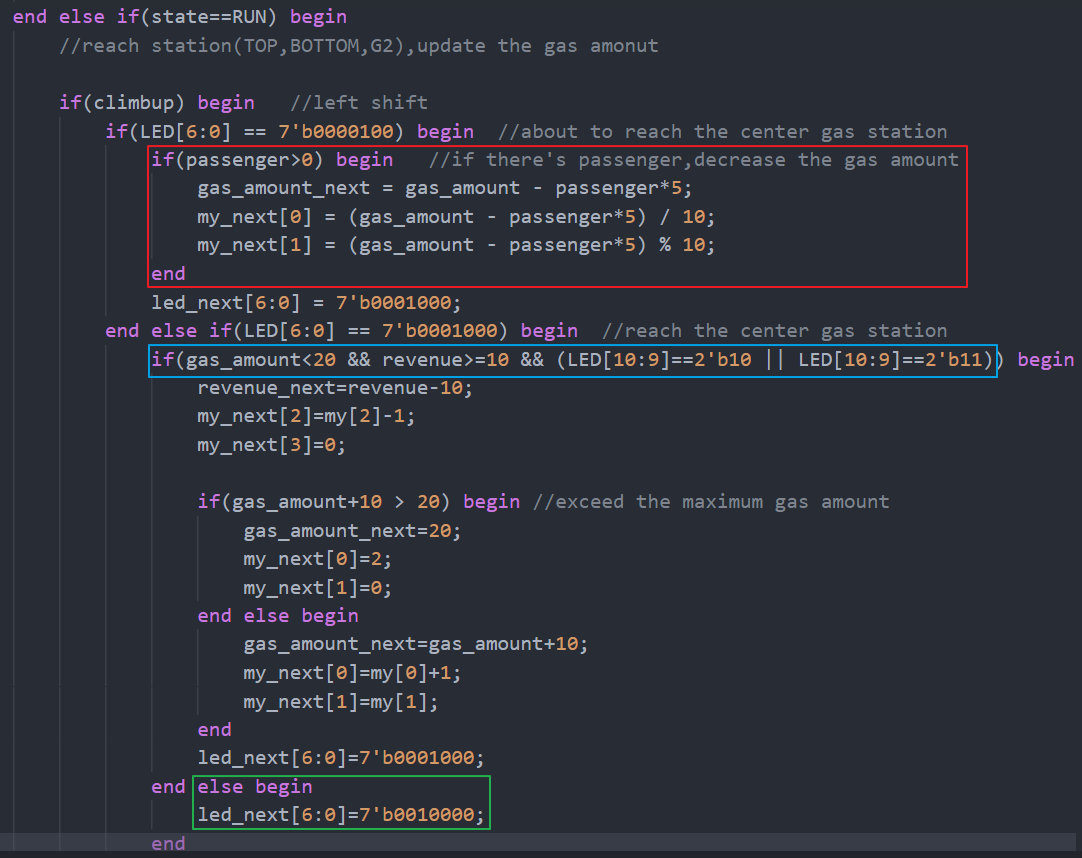
在完成上車後以及尚未收錢的時候收取費用，這部分比較單純，僅須注意錢超過最大值90的情況，利用/,%真的方便許多。

(圖3:在BOTTOM中處理加油)



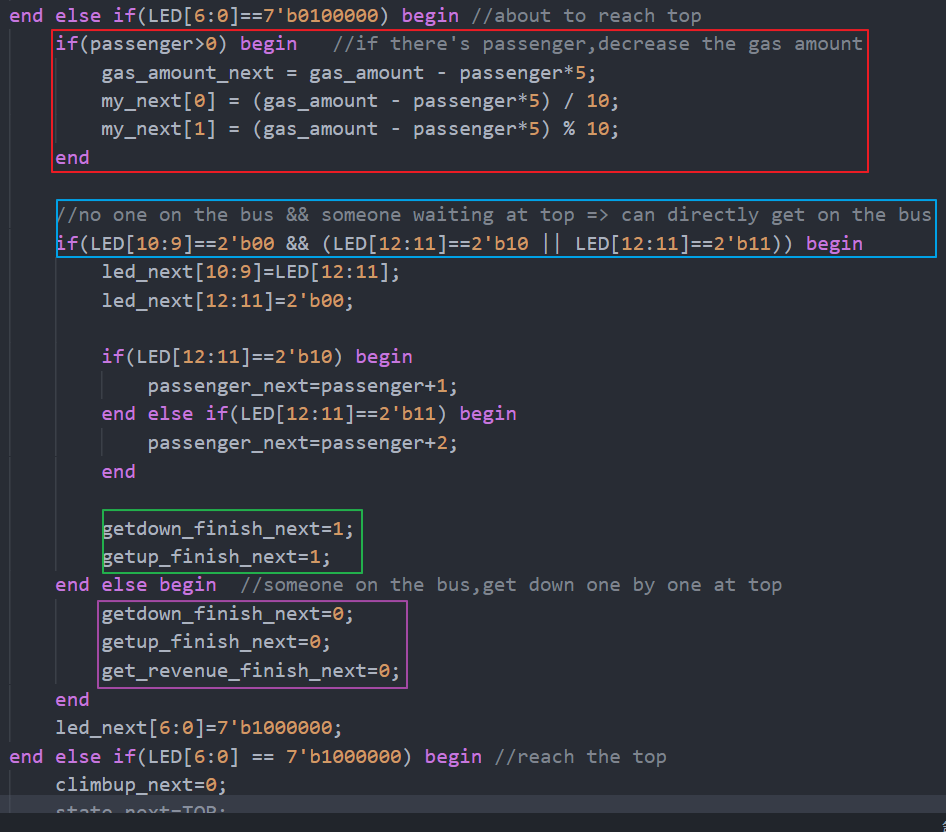
紅框部分是收完錢才需要加油，藍框部分的條件讓我想了許久，得出的結論是只有在油沒滿&&有錢&&有乘客的時候才需要加油，其他狀況都不用(沒錢、沒乘客…)，須注意油量超過最大值20的情況，加滿油或是不需加油的情況就是綠框部分，準備進入RUN以及重置所有flag。

(圖4:RUN的第一部分)



紅框部分是在準備進到中間加油站的時候要檢查乘客的數量，如果有乘客就根據乘客數去扣油量，my\_next[0]顯示油量的十位數，my\_next[1]顯示個位數。藍框部分跟上面的解釋一樣，只有在油沒滿&&有錢&&有乘客的時候才需要加油，其他狀況都不用(沒錢、沒乘客…)，如果油加滿了或是不用加油，公車就繼續往前(綠框部分)

(圖5:RUN的第二部分)



快要到山上的時候，需要檢查車上目前的人數(LED[10:9])以及山頂上的等車人數(LED[12:11])，因為如果車上沒人而且有人在山上等車(也就是滿足藍框的條件)，等車的人可以在公車進站的那一瞬間上車，但是如果車上有人就必須等公車進站後(進入TOP state)再依序下車(紫色部分)。需要特別小心的就是flag的處理，如果進站前車上沒人乘客可以馬上上車，上車完就必須把上下車的flag設為1(這是因為我的TOP,BOTTOM state一開始會根據flag的狀態進行上下車，如果把綠框部分上下車的flag設為0的話，那麼剛上公車的乘客會因為符合TOP,BOTTOM state下車的if條件式(乘客數量>0 && 尚未完成下車)而被趕下去)。

(圖6:RUN的第三部分)

一張含有 文字, 裝置, 儀錶, 控制台 的圖片

自動產生的描述

到了山頂就將climbup的方向反轉，下面的else是公車開上山的動作。

(圖7: STANDBY)

一張含有 文字, 室內, 螢幕擷取畫面 的圖片

自動產生的描述

STANDBY只需要維持所有變數以及監看兩邊的公車站有沒有人突然冒出來，如果突然有人出現，則需要根據公車目前的所在位置決定該切換到哪個state。例如第一種情況: 公車目前停在山下且有人在山下等，那這時候就要切換到BOTTOM讓等車的乘客上車並付錢。

1. 學到的東西與遇到的困難
2. 公車到站時處理下車、上車、付錢、加油的順序

我花了一些時間仔細看demo影片後歸納出有這4個動作，一開始有想說讓加油自己獨立出來一個state，但這樣每次加油的時候都要切換state有點麻煩。所以後來我將這四個動作都寫在公車到站的state(BOTTOM, TOP)裡面。

1. 處理完成下車、上車、付錢、加油的flag

為了處理這四個動作的順序，我設了3個flag，分別是完成下車、完成上車、完成付錢。

但flag一多，要處理它們就變得有些棘手，我打算在BOTTOM,TOP state將每個flag分開判斷與處理，不要寫出很深的巢狀if，這樣code會冗長不易懂導致難以debug。

下車的時候檢查乘客數量以及是否完成下車

上車的時候必須是已完成下車以及尚未完成上車

收錢的時候必須是已完成上車並且尚未完成收錢

加油的時候必須是已經收完錢的狀態

(參考1.實作過程的圖1,2,3中紅框部分)

1. 想對老師或助教說的話

Verilog越寫越順手，lab5,6都寫很快。

雖然在這次lab的spec中沒有說FSM有哪幾個state，讓我在設計FSM的時候想好久，前前後後想了三個版本的state diagram，不過我在設計的過程中FSM所需的state越來越少(7->5->4個state)，應該也是一件好事吧。

願望: 希望以後的spec中已經寫有哪幾個state了，這樣寫起來比較開心。