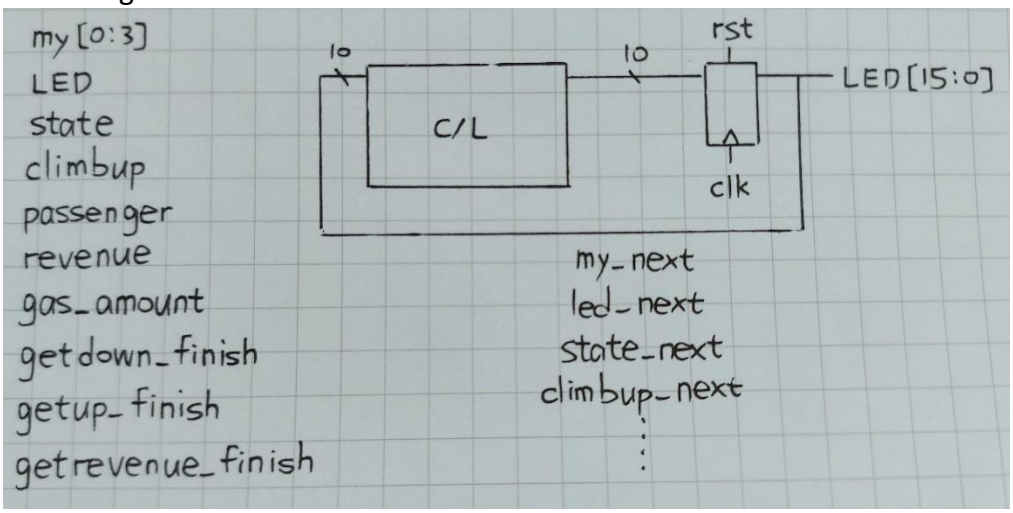
Lab 6

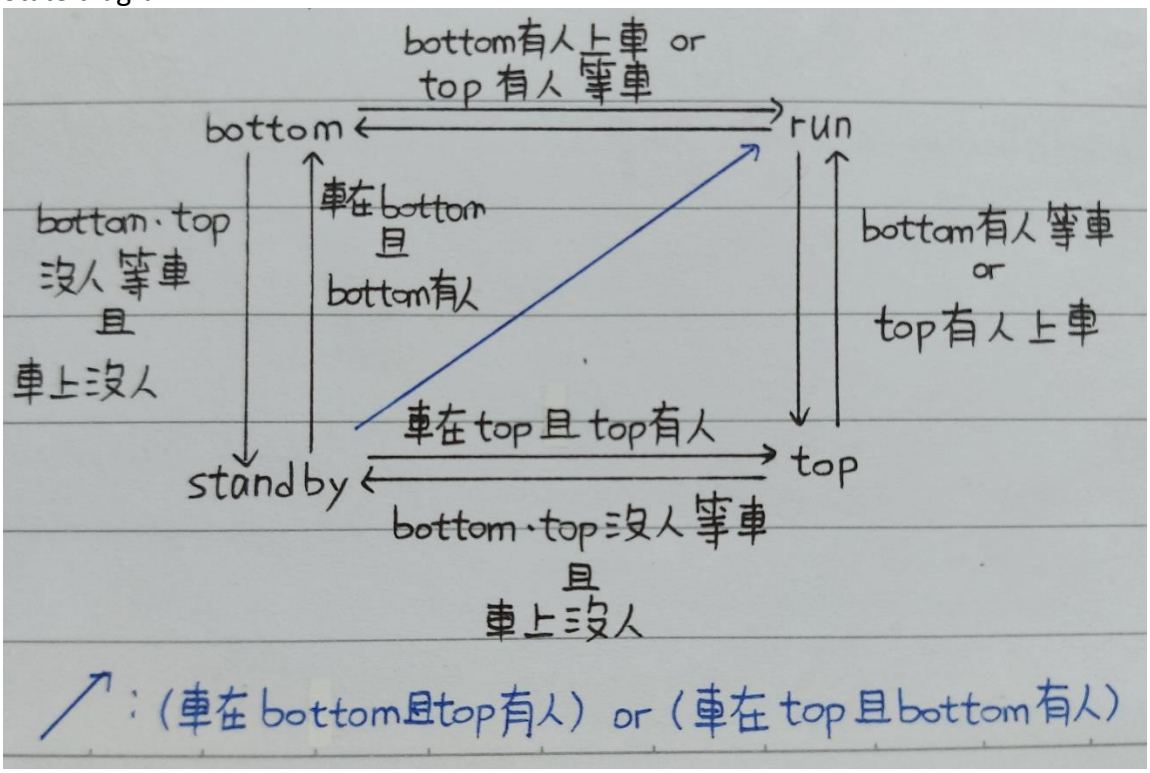
學號: 109062318 姓名: 簡弘哲

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)

```
if(state==BOTTOM) begin
    climbup_next=1;
    //get off the bus
    if(passenger>0 && !getdown_finish) begin   //have passenger => get down one by one
        led_next[10:9] = {LED[9], 1'b0};    //left shift

        passenger_next = passenger - 1; //update passenger count
    end else begin  //no passenger => finish get down
        getdown_finish_next=1;
    end

    //get on the bus
    if(getdown_finish && !getup_finish) begin
        if(LED[15:14]==2'b00 && LED[12:11]==2'b00) begin
            state_next=STANDBY;
        end else begin
            led_next[10:9] = LED[15:14];    //get on the bus at once
            led_next[15:14] = 2'b00;

            if(LED[15:14]==2'b11) begin //update passenger count
                passenger_next = passenger + 2;
            end else if(LED[15:14]==2'b10) begin
                passenger_next = passenger + 1;
            end

            getup_finish_next=1;
        end
    end
```

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)

```
//get revenue
if(getup_finish && !get_revenue_finish) begin
    if(revenue + passenger*30 > 90) begin   //beware of $ exceeding the maximum amount
        revenue_next=90;
        my_next[2]=9;
        my_next[3]=0;
    end else begin
        revenue_next=revenue + passenger*30;
        my_next[2]=(revenue + passenger*30) / 10;
        my_next[3]=0;
    end
    get_revenue_finish_next=1;
end
```

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)

```
//fueling
if(get_revenue_finish) begin
    if(gas_amount<20 && revenue>=10 && (LED[10:9]==2'b10 || LED[10:9]==2'b11)) begin
        revenue_next=revenue-10;
        my_next[2]=my[2]-1;
        my_next[3]=0;

        if(gas_amount+10 > 20) begin //exceed the maximum gas amount
            gas_amount_next=20;
            my_next[0]=2;
            my_next[1]=0;
        end else begin
            gas_amount_next=gas_amount+10;
            my_next[0]=my[0]+1;
            my_next[1]=my[1];
        end
    end else begin  //finish fueling => start to run
        state_next=RUN;

        //reset all flags before change state
        getdown_finish_next=0;
        getup_finish_next=0;
        get_revenue_finish_next=0;
    end
end
```

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)

```
end else if(state==RUN) begin
    //reach station(TOP,BOTTOM,G2),update the gas amonut

    if(climbup) begin   //left shift
        if(LED[6:0] == 7'b0000100) begin  //about to reach the center gas station
            if(passenger>0) begin   //if there's passenger,decrease the gas amount
                gas_amount_next = gas_amount - passenger*5;
                my_next[0] = (gas_amount - passenger*5) / 10;
                my_next[1] = (gas_amount - passenger*5) % 10;
            end
            led_next[6:0] = 7'b0001000;
        end else if(LED[6:0] == 7'b0001000) begin  //reach the center gas station
            if(gas_amount<20 && revenue>=10 && (LED[10:9]==2'b10 || LED[10:9]==2'b11)) begin
                revenue_next=revenue-10;
                my_next[2]=my[2]-1;
                my_next[3]=0;

                if(gas_amount+10 > 20) begin //exceed the maximum gas amount
                    gas_amount_next=20;
                    my_next[0]=2;
                    my_next[1]=0;
                end else begin
                    gas_amount_next=gas_amount+10;
                    my_next[0]=my[0]+1;
                    my_next[1]=my[1];
                end
                led_next[6:0]=7'b0001000;
            end else begin
                led_next[6:0]=7'b0010000;
            end
```

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)

```
end else if(LED[6:0]==7'b0100000) begin //about to reach top
    if(passenger>0) begin   //if there's passenger,decrease the gas amount
        gas_amount_next = gas_amount - passenger*5;
        my_next[0] = (gas_amount - passenger*5) / 10;
        my_next[1] = (gas_amount - passenger*5) % 10;
    end

    //no one on the bus && someone waiting at top => can directly get on the bus
    if(LED[10:9]==2'b00 && (LED[12:11]==2'b10 || LED[12:11]==2'b11)) begin
        led_next[10:9]=LED[12:11];
        led_next[12:11]=2'b00;

        if(LED[12:11]==2'b10) begin
            passenger_next=passenger+1;
        end else if(LED[12:11]==2'b11) begin
            passenger_next=passenger+2;
        end

        getdown_finish_next=1;
        getup_finish_next=1;
    end else begin  //someone on the bus,get down one by one at top
        getdown_finish_next=0;
        getup_finish_next=0;
        get_revenue_finish_next=0;
    end
    led_next[6:0]=7'b1000000;
end else if(LED[6:0] == 7'b1000000) begin //reach the top
    climbup_next=0;
    state_next=TOP;
```

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)

```
end else if(LED[6:0] == 7'b1000000) begin //reach the top
    climbup_next=0;
    state_next=TOP;
end else begin
    led_next[6:0]={LED[5:0],1'b0};  //left shift
end
```

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)

```
end else if(state==STANDBY) begin
    //maintain everything
    //check both bus stop
    if(LED[6:0]==7'b0000001 && (LED[15:14]==2'b10 || LED[15:14]==2'b11)) begin //bus at bottom && someone waiting at bottom
        state_next=BOTTOM;
    end else if(LED[6:0]==7'b1000000 && (LED[12:11]==2'b10 || LED[12:11]==2'b11)) begin   //bus at top && someone waiting at top
        state_next=TOP;
    end else if((LED[6:0]==7'b0000001 && (LED[12:11]==2'b10 || LED[12:11]==2'b11)) ||
                (LED[6:0]==7'b1000000 && (LED[15:14]==2'b10 || LED[15:14]==2'b11)) ) begin  //bus at bottom,someone waiting at top ||
                                                                                            //bus at top,someone waiting at bottom
        state_next=RUN;
    end else begin
        state_next=STANDBY;
    end
end
```

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。

2. 學到的東西與遇到的困難

(1) 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。

(2) 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)

3. 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。