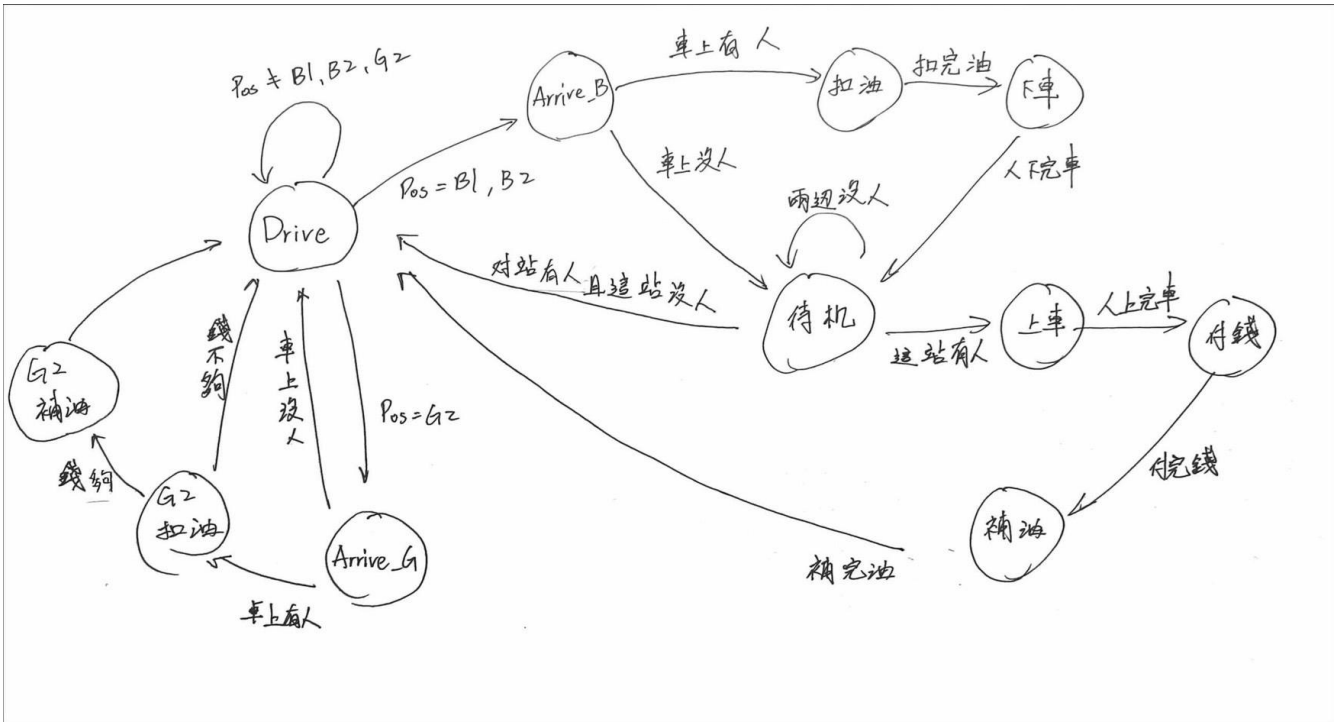
Lab 6

學號: 109062202 姓名: 陳禹辰

1. 實作過程

這次 LAB 相較於前幾次比較沒有規定說要有那些 state，所以如果一開始直接寫就會發現條件判斷很多，如果不先設計 FSM 會寫得很複雜而且比較難下手。所以我就先依照題意設計了自己的 FSM，我主要是分成 開車(DRIVE)、到達(ARRIVE) 、扣油、下車、待機、上車、付錢、補油八個 state，然後到達、扣油跟補油又分成是在 G2 加油站還是 B1、B2 車站。下面是我 FSM 的圖。



平常行進就是在 DRIVE，如果 bus pos 在 G2 就會進到 ARRIVE_G，判斷要不要扣油(如果車上有人要扣油，車上沒人則不用)、要不要補油(錢夠就要補不購則不用)，然後再回到 DRIVE 繼續行進。而若是 bus pos 在 B1 或 B2 則是進到 ARRIVE_B 接著看車上有沒有人，若是有人就要先扣油再讓人下車然後待機，車上沒人則是直接待機。接著判斷公車需不需要開，兩邊都沒人的狀況是維持在待機，而若是對站有人這站沒人則開車過去接人，其餘情況就是接這站的人上車開往對站。而若是這站有人要上車，就先上車再付錢跟補油，最後再出發。

把 FSM 設計完之後我就是照著我的 FSM 打，因為前幾次已經練習蠻多了所以設計完 FSM 後就比較好進行了。

```
/*------------------------------LED-------------------------------------*/
/*----------------------------KEYBOARD----------------------------------*/
/*------------------------------BUS-------------------------------------*/
/*------------------------------BCD-------------------------------------*/
```

實作時，我分成四個部分來處理，分別是 LED(處理 LED 該亮什麼燈)、KEYBOARD(處理鍵盤的操作)、BUS(處理 FSM 裡面各個數值的更新，然後在傳給 LED 跟 BCD 讓他知道該顯示什麼)、

BCD(處理 7-segement 的顯示)，接著分別講述裡面的東西。

首先是 LED，因為 B1 跟 B2 的人數是一按鍵盤就要更新所以我用的 clk 是跟鍵盤一樣的 clk，而 bus 的移動跟人下車時的速度就是跟 FSM 的速度是一樣的所以我就用不同於 B1 跟 B2 的 clk。主要就是用 BUS 裡面更新的數值來決定 LED 上要顯示什麼東西。

```
// LED refresh  (need LED)
always @(posedge clk_bus or posedge rst) begin
    if (rst) begin
        LED[10:0] <= 11'b000_0000_0000;
    end else begin
            // [8:7]   [6:0]
        LED[10:0] <= {Bus_led, 2'b0, Pos_led};
    end
end
assign clk_bus = (Bus_state == GET_ON) ? clk : clk_machine;

always @(posedge clk or posedge rst) begin
    if (rst) begin
        LED[15:11] <= 7'b000_0000;
    end else begin
                //[15:14] [13]  [12:11]
        LED[15:11] <= {B1_led, 1'b0, B2_led};
    end
end
```

KEYBOARD 的部分我就是參考 smaple 的做法進行，就是判斷最後按下的案件是不是 right1,2 然後決定 B1 跟 B2 的人數要不要增加。

BUS 則是處理我 FSM 的數值更新，除了 B1 跟 B2 的人數我是用 clk 其他都是與 FSM 的 clk 相同。

```
always @(posedge clk_FSM or posedge rst) begin
    if (rst) begin
        Bus_pos <= 0;
        Bus_state <= WAITING;
        Bus_dir <= 0;
        revenue <= 0;
        gas_unit <= 0;
    end else begin
        Bus_pos <= Bus_pos_next;
        Bus_state <= Bus_state_next;
        Bus_dir <= Bus_dir_next;
        revenue <= revenue_next;
        gas_unit <= gas_unit_next;
    end
end
assign clk_FSM = (Bus_state == GET_ON) ? clk : clk_machine;
```

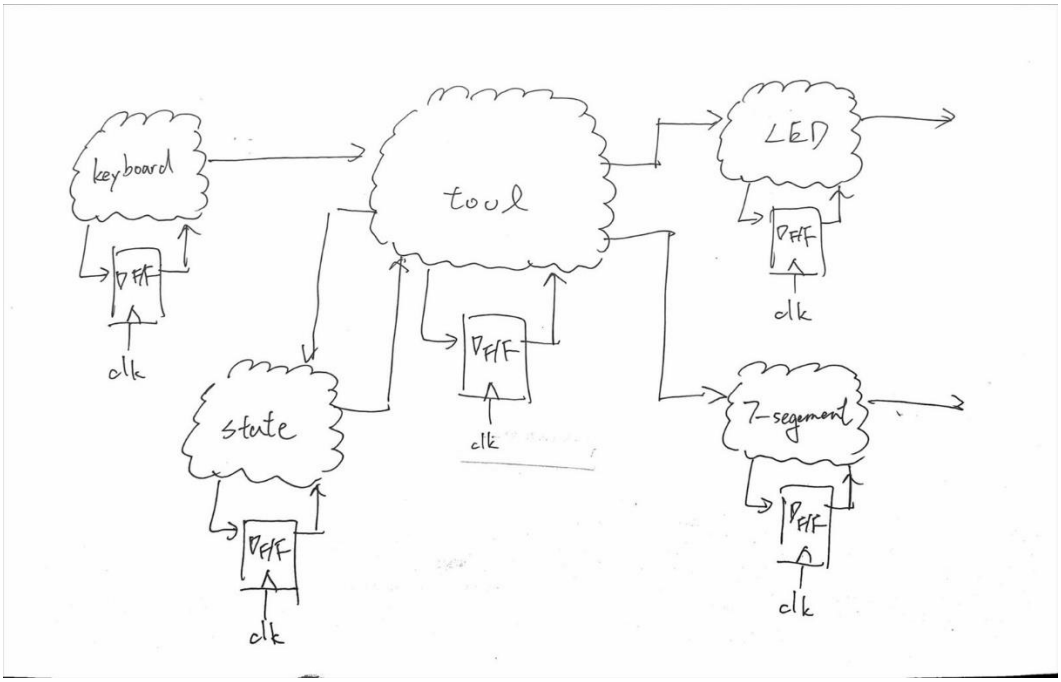
```
always @(posedge clk_people or posedge rst) begin
    if (rst) begin
        Bus_people <= 0;
    end else begin
        Bus_people <= Bus_people_next;
    end
end
assign clk_people = (Bus_state == GET_ON) ? clk : clk_machine;

always @(posedge clk or posedge rst) begin
    if (rst) begin
        B1_people <= 0;
        B2_people <= 0;
    end else begin
        B1_people <= B1_people_next;
        B2_people <= B2_people_next;
    end
end
```

然後就是 state change 跟各個需要在不同 state 做更新的數值，就是看在哪個 state 要做什麼事，然後什麼樣的情況會造成 state change，大致上的方法就跟前幾次 lab 做 FSM 的方法差不多，就不特別贅述。

BCD 則是處理 7-segement 要顯示的東西，就是當前的收入與油的總量。

```
always @(posedge clk_light) begin
    case (DIGIT)
        4'b1110: begin
            value <= revenue / 7'd10;
            DIGIT <= 4'b1101;
        end
        4'b1101: begin
            value <= gas_unit % 5'd10;
            DIGIT <= 4'b1011;
        end
        4'b1011: begin
            value <= gas_unit / 5'd10;
            DIGIT <= 4'b0111;
        end
        4'b0111: begin
            value <= revenue % 7'd10;
            DIGIT <= 4'b1110;
        end
        default: begin
            value <= revenue % 7'd10;
            DIGIT <= 4'b1110;
        end
    endcase
end
```

這是大致上的 block diagram，state 的會把現在是什麼 state 傳給 tool 判斷哪些東西要做更新，然後再根據 tool 的值來判斷 state 要不要改變，然後從 tool 裡的數值決定 LED 跟 7-segement 的顯示要為何。



2. 學到的東西與遇到的困難

這次遇到最大的困難就是一開始有點不太知道從何下手，因為有鍵盤是之前沒處理過的，然後 spec 上對於 FSM 各個 state 的劃分也比較不明確，跟室友討論了一下後才開始設計 FSM 然後依照設計的 FSM 慢慢完成。不過也因此更了解在開始做 LAB 之前，FSM 的設計跟規劃要如何做的重要性，我覺得這對於之後進行 final project 肯定是個很好的練習。

因為 FSM 已經練習很多次了，所以基本上沒遇到什麼太大的問題，基本上有 bug 也只要稍微 trace code 就可以找出問題所在了，比較需要特別思考的應該只有跟 clk 相關的問題了，因為有時候預想的操作都有進行，可是就是時間點不對，所以之後就要特別注意這方面的東西。

3. 想對老師或助教說的話

謝謝助教跟教授，然後希望下次期中考的題目可以有較充裕的時間。

每次打完扣會出現的情況。。。。